De Montfort University
Faculty of Technology
Mechatronics Research Centre

A New Approach to Systems Integration in the Mechatronic Engineering Design Process of Manufacturing Systems

A dissertation submitted in partial fulfilment
of the requirements for the degree of Doctor of Philosophy

Name: Malte Proesser

Registration Date: 01/07/2008
Submission Date: 03/06/2014

First Supervisor: Professor Philip Moore [a]
Second Supervisor: Chi-Biu Wong
Second Supervisor: Professor Ulrich Schmidt [b]

FALMOUTH UNIVERSITY

[a] Falmouth University
Academy For Innovation & Research (AIR)

Technische Hochschule Ingolstadt

[b] Technische Hochschule Ingolstadt
Center for Applied Research (ZAF)

Abstract

Creating flexible and automated production facilities is a complex process that requires high levels of cooperation involving all mechatronics disciplines, where software tools being utilised have to work as closely as their users. Some of these tools are well-integrated but others can hardly exchange any data. This research aims to integrate the software systems applied by the mechatronic engineering disciplines to enable an enhanced design process characterised by a more parallel and iterative workflow.

This thesis approaches systems integration from a data modelling point of view because it sees information transfer between heterogeneous data models as a key element of systems integration. A new approach has been developed which is called *middle-in* data modelling strategy since it is a combination of currently applied top-down and bottom-up approaches. It includes the separation of data into *core design data* which is modelled top-down and *detailed design data modules* which are modelled bottom-up.

The effectiveness of the integration approach has been demonstrated in a case study undertaken for the mechatronic engineering design process of body shop production lines in the automotive industry. However, the application of the middle-in data modelling strategy is not limited to this use case: it can be used to enhance a variety of system integration tasks.

The middle-in data modelling strategy is tested and evaluated in comparison with present top-down and bottom-up data modelling strategies on the basis of three test cases. These test cases simulated how the systems integration solutions based on the different data modelling strategies react to certain disturbances in the data exchange process as they would likely occur during industrial engineering design work. The result is that the top-down data modelling strategy is best in maintaining data integrity and consistency while the bottom-up strategy is most flexibly adaptable to further developments of systems integration solutions. The middle-in strategy com-

bines the advantages of top-down and bottom-up approaches while their weaknesses and disadvantages are kept at a minimum. Hence, it enables the maintenance of data modelling consistency while being responsive to multidisciplinary requirements and adaptive during its step-by-step introduction into an industrial engineering process.

Acknowledgements

In the following, I would like to thank a number of great people who have contributed to this research. First I would like to express my sincere gratitude to my supervisors Professor Philip Moore from Falmouth University, Chi-Biu Wong from De Montfort University and Professor Ulrich Schmidt from Technische Hochschule Ingolstadt. Their guidance throughout the project especially their honest and constructive feedback contributed decisively to this thesis.

I would like to thank the the Tools and Equipment Building Division of the AUDI AG in Ingolstadt for funding this research. I am very grateful to all my colleagues in this division for the productive collaboration and the valuable discussions. I am especially obliged to Frank Herzog who was head of the Construction Department of the funding division. He had great confidence in this work and created the conditions for the research project. Additionally, I thank Dr. Michael Ehrenstrasser for his support in working out the research thread. Last but not least, I am indebted to Jürgen Wilhelmy for his unbreakable faith in me and his steady encouragements.

This research project was undertaken at the Centre for Applied Research at the Technische Hochschule Ingolstadt and I thank all students who contributed to this thesis. I am particularly obliged to all my peers at the centre, namely, Dr. Stefanie Angerer, Dr. Andreas Hermann, Dr. Stephan Matzka and Christoph Strassmeir, for sharing the highs and lows in research.

My earnest gratitude goes to my family and my friends for their constant support and encouragement, offering retreat when needed. Especially, I offer my sincere thanks to Dr. Gunnar Hartung for his proof-reading and suggested improvements in the final stage of writing this thesis.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

API Application Programming Interface

ASB Automation Service Bus

AutomationML Automation Modelling Language

CAD Computer Aided Design

CAE Computer Aided Engineering

CAEX Computer Aided Engineering eXchange

CIM Computer Integrated Manufacturing

CIMOSA CIM Open System Architecture

COLLADA COLLAborative Design Activity

COM Component Object Model

CORBA Common Object Request Broker Architecture

CSV Comma-Separated Values

EDB Engineering Database

EEC Eplan Engineering Center

EKB Engineering Knowledge Base

ESB Enterprise Service Bus

ESPIRIT European Strategic Program on Research in Information Technology

HMI Human Machine Interface

ICAM Integrated Computer-Aided Manufacturing

IGES Initial Graphics Exchange Specification

JMS Java Message System

MEDEIA Model-Driven Embedded Systems Design Environment for the Industrial Automation Sector

MOM Message-Oriented Middleware

MQ Message Queue

NGCCAS Next-Generation Collaboration and Configurable Automation Systems

PACT Palo Alto Collaborative Testbed

PDM Product Data Management

PLC Programmable Logic Controller

PLM Product Lifecycle Management

STEP Standard for the Exchange of Product Data

VR Virtual Reality

XML eXtensible Markup Language

1. Introduction

This dissertation presents a new approach to systems integration in the mechatronic engineering design process of manufacturing systems. It focuses on the data modelling aspects of systems integration which makes it independent from the implementation method and allows the identification of crucial process steps for the successful introduction and efficient application of digital manufacturing.
This chapter introduces the research by providing background information, stating the aims and objectives as well as outlining the scope of this research.

1.1. Research Background

The design of manufacturing systems is constantly influenced by product and process driven requirements. The product driven requirements are related to changes of the product produced on a manufacturing system and the process driven requirements are related to the production process and the manufacturing facilities.
A major trend influencing the product driven requirements on manufacturing systems is the increasing demand for personalised and custom build products. This trend requires flexible and agile manufacturing systems able to flexibly produce a changing variety of products in any number and in any sequence. Lack of agility and responsiveness to market changes were identified as two of the existing problems with automation industry (Haq et al., 2010). Manufacturing machine systems, therefore, need to respond to changes in products during the shortest time and at minimum cost (Ng, 2003).
The process driven requirements include the demand for short commissioning and ramp-up time with the smallest possible personnel placement. In addition, manufacturing system operators require their production facilities to need little maintenance and to have comprehensive diagnosis functionality to control and enhance their production process.
Both, product and process driven requirements can only successfully be met through a close cooperation between mechanical and electrical engineering as well as software development in a mechatronic engineering design process.

1.1.1. Data Exchange as Basic Prerequisite for Successful Mechatronic Engineering Design

Manufacturing systems engineering design processes are often seen as sequential work in mechanical, electrical and software engineering disciplines. However, in the design of 21st century manufacturing systems engineers face major challenges. These challenges include realising manufacturing systems in decreasing time. Moreover, the functional complexity of the manufacturing systems is increasing in such a way that the engineering disciplines are more and more functionally linked with each other. Mechanical design decisions have a big impact on the electrical and software design. A functional design problem can often be solved by an improved mechanical design as well as with an improved electrical design. Both solutions might be linked to different efforts or additional costs which require the engineering disciplines to communicate in order to find the optimal solution for the problem.

The need to complete manufacturing systems in decreasing time as well as the need for designers to find solutions for design problems in an interdisciplinary way is leading to a more and more parallelised work in the engineering disciplines. This parallel work has a deep impact on the way engineers exchange information. A sequential approach to work implies an exchange of final engineering results. In contrast to this, a parallel workflow necessitates a continuous exchange of provisional engineering results. Since engineering design work today is mostly undertaken with the support of software tools, there is an increasing need for these software tools to become more and more integrated to seamlessly exchange provisional engineering results. This makes the seamless data exchange between the specialised software tools employed by the different engineering disciplines a basic prerequisite for a successful mechatronic engineering design process.

1.1.2. Current Status of Technology in Industry

In many cases, the tool landscape in the engineering design process of manufacturing systems is today still very heterogeneous and far from being seamlessly integrated. Every engineering discipline uses software tools suited to its specific needs. These software tools have been developed independently of each other, thus they are frequently unable to exchange any data. However, there are several approaches for setting up an integrated system including all mechatronics engineering data as illustrated in Figure 1.1.
During the last few years, great efforts have been undertaken in both science and

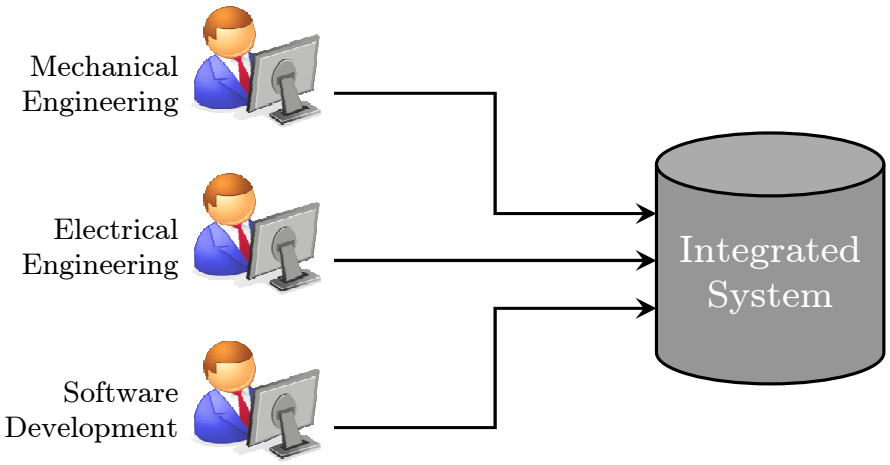


Figure 1.1.: Basic model of systems integration

industry to develop an integrated system, including all mechatronics disciplines to share their engineering data. Comprehensive software suites have been developed incorporating tools for the design tasks of every engineering discipline. The promised advantages of an integrated system are apparent and thus are widely accepted in research and industry. However, the application of a fully integrated system has not yet become a reality in the majority of industrial companies. One reason for this is that an integrated tool suite is intended to replace the present tool landscape and thus its introduction to an industrial engineering process is connected with high technical and economic risks.

The integrated system itself is of less interest to this research; rather the focus is on the way the information from the different disciplines is incorporated. With reference to Figure 1.1, the focus of this research is the arrows leading from the disciplines to the integrated system. This includes the identification of the required preparation for successfully introducing and efficiently applying an integrated system in a mechatronic engineering design process. This means an in-depth assessment of the current engineering process in terms of *who needs what data from whom at what time and in which form*. To answer this question, the engineers need to use a common nomenclature for their designs, including all components and subcomponents. In terms of software systems, a common nomenclature is a normal means of storing information in different data models. Therefore this research focuses on data modelling as a key enabling factor for data exchange and efficient collaboration in a mechatronic engineering design process.

1.2. Research Aim and Objectives

The principal aim of this study is to advance the mechatronic engineering design process of manufacturing systems through enabling consistent modelling of the processed data for a seamless data exchange among the software tools involved. Such an aim can be further refined into the following specific objectives:

1. Explore current approaches to systems integration and data sharing with regard to their ability to handle the major challenges in this domain. Identify the essential conditions for the efficient incorporation of these approaches in an industrial environment.

2. Analyse the influence of the data modelling strategy on the efficiency of data exchange in the mechatronic engineering design process of manufacturing systems.

3. Introduce a new strategy to model engineering design data for enhanced interoperability of heterogeneous software tools ready to be implemented in industrial processes.

4. Evaluate the proposed data modelling strategy by means of a real industrial-scale implementation and assess the transferability of the approach to other domains.

1.3. Scope of the Dissertation

This dissertation is derived from a three year research project undertaken at the Center for Applied Research (ZAF) of the Technische Hochschule Ingolstadt. The project was commissioned and funded by the Tools and Manufacturing Equipment Division of the AUDI AG. Among other duties, this division is tasked with the planning, design and commissioning of body shop production lines. However, the applicability of the research results is not limited to mechatronic engineering of body shop production lines. The results are, instead, applicable to multidisciplinary engineering design processes of highly automated, single-batch and custom built, special purpose machines or production lines as employed in many industries, for example the food and textile industries. Such a multidisciplinary engineering design process generally involves the mechatronics disciplines: mechanical and electrical engineering as well as software development. In terms of manufacturing system design, the software development includes designing applications for industrial controls such as

PLCs and robots. In addition to this, the design of manufacturing systems incorporates disciplines such as factory layout design, hydraulics or pneumatics design as well as simulation and virtual commissioning.
This dissertation includes a review of current literature and the development of a new data modelling approach for enhanced collaboration in interdisciplinary engineering design processes. This new approach was implemented and evaluated in a case study.

1.3.1. Literature Review and Classification of Data Modelling Strategies

The literature review starts by assessing the challenges of effective collaboration between interdisciplinary engineering teams. Other publications reviewed deal with the technical aspects of interdisciplinary engineering collaboration including approaches aiming to facilitate seamless data exchange between the applied software systems. System integration is then analysed with regard to data modelling, where data exchange is considered as information transfer between different data models. Top-down and bottom-up strategies are identified as two general approaches to data modelling. A middle-in data modelling strategy is presented as a combination of the top-down and bottom-up approaches. The three data modelling strategies are compared with respect to their abilities to face the major challenges connected with data modelling.

1.3.2. Data Modelling Approach to Enhance Data Exchange

A new approach to systems integration based upon the middle-in data modelling strategy is presented. It is commonly described by means of set theory, interpreting engineering data from the mechatronic disciplines as overlapping sets of information. These theoretical considerations are transferred to real data from an existing mechatronic engineering design process to illustrate the characteristics of discipline specific data and discipline overlapping data.

1.3.3. Case Study and Evaluation

The current research includes a prototypical implementation of a new middle-in data modelling strategy for systems integration. This implementation was undertaken in the engineering design process of the body shop production lines of the AUDI AG's tooling and manufacturing equipment division. The case study focuses on the in-

terface between mechannical and electrical engineering, in particular the exchange of production line layout data and penumatic design data.
The benefits of the middle-in data modelling strategy are qualitatively evaluated with the aid of three test cases simulating the reaction of systems integration solutions based on different data modelling strategies on certain disturbances as they typically occur in industrial engineering design processes. This evaluation assesses the capacity of the middle-in data modelling strategy, regarding three major challenges in systems integration, namely, its ability to assure data model consistency, the conditions for gaining broad user acceptance and the possibility of being introduced to an industrial engineering design process in stages.

1.4. Dissertation Organisation

Chapters 2 and 3 deal with the current state of systems integration to support collaborative, interdisciplinary engineering design processes. Chapter 2 presents a broad literature review while Chapter 3 analyses systems integration approaches from a modelling point of view, including a classification according to their underlying data modelling strategy.
Chapter 4 is the core chapter, presenting the new middle-in data modelling approach to systems integration. The following two chapters are devoted to the implementation of the middle-in data modelling strategy: Chapter 5 presents the fundamental considerations for a successful implementation and Chapter 6 outlines the case study of implementing a prototypic systems integration solution in the engineering design process of body shop production lines.
Chapter 7 evaluates the new approach and the case study on the basis of three test cases and finally Chapter 8 draws conclusions of this thesis and offers an outlook on future work.

2. Literature Review

The literature review provides an overview on the current status of research in the field of systems integration from a general perspective on engineering disciplines collaboration as well as specific focus on implementation methods for systems integration. Since systems integration aims to improve engineering discipline collaboration, this literature review starts with assessing problems and requirements as well as current approaches to improving engineering discipline collaboration.
This review chapter builds upon studies by Bakis et al. (2007), MEDEIA Consortium (2008) and Shen et al. (2008). MEDEIA Consortium (2008) presents an overview on the field with a focus on model driven design in embedded systems. Description languages for interdisciplinary data exchange, which is also relevant to this research, are also included. However, the reviews from Bakis et al. (2007) and Shen et al. (2008) are related to distributed product data sharing environments and systems integration for collaboration in construction which also correspond to the focus of this research.

2.1. Collaboration for Interdisciplinary Engineering

Assessing collaboration of engineering disciplines on a general level is the starting point of this literature review because any systems integration process aims at improving their interdisciplinary collaboration. Efficient collaboration with other companies and suppliers as one of the core competences of manufacturing engineering, systems integration and digital manufacturing would allow for focusing more on these competencies (Chryssolouris et al., 2009). This section assesses current problems and requirements for efficient interdisciplinary engineering collaboration and summarises current approaches to improving the process.

2.1.1. Current Problems and Requirements

While mechatronic systems indeed incorporate elements constructed by different engineering disciplines and computer science, the actual cooperation during construction is less developed. One of the most prevalent problems in current industrial

development and even research approaches is the lack of integration with different disciplines, namely mechanical, electrical and control engineering (Schafer and Wehrheim, 2007). The necessity of ensuring consistency among engineering activities and the data supporting the applied software becomes one of the crucial points of the mechatronic engineering process (Lüder et al., 2010). One reasons for inconsistency among engineering activities is the lack of clearly defined responsibilities and transparent data ownership (Drath and Barth, 2011).
Previous studies have not considered the integration and sharing of product engineering knowledge in the life cycle of collaborative product design and process development. Instead, previous studies focused primarily on knowledge integration and sharing for a product structure (Chen, 2010). Thus, additional approaches are required to improve the methodologies for collaborative interdisciplinary engineering design.

2.1.2. Approaches to Improved Interdisciplinary Collaboration

To improve interdisciplinary collaboration in engineering design processes, the well known mechatronic paradigm has been adapted to manufacturing systems during the past decade (Lüder et al., 2010). This adaptation includes approaches inspired by the component-based paradigm, e.g. Adolfsson et al. (2002), Ng (2003). Their studies propose a highly integrated design, simulation and programming environment that can be applied to realise agility in manufacturing machine systems. Design, programming, testing and verification of the manufacturing machine systems are facilitated by first being simulated in a virtual environment and then seamlessly transferred to the distributed control system environment for controlling real machines/devices. Similarly Haq et al. (2010) aims to assist in the creation of a new engineering environment to build and configure machines from reusable smart modules to support concurrent engineering between product, process and control engineering. He proposed and developed a new realisation approach called Next-Generation Collaboration and Configurable Automation Systems (NGCCAS).
In the early 1990s, in field of artificial intelligence, Neches et al. (1991) established the term *ontology* for the formal description of knowledge. Cutkosky et al. (1993) transferred this idea to the field of collaborative engineering design processes by developing the Palo Alto Collaborative Testbed (PACT). Other researchers, e.g. Chen (2010), have improved this approach. He applied ontology-based knowledge integration methods to develop ontology-based knowledge integration and a sharing mechanism for the collaborative process of moulding product design and process development.

Other approaches aim to improve interdisciplinary engineering by introducing collaboration software. For example Drath and Barth (2011) pursues the concepts of Data Ownership, Collaboration Objects and Data Exchange Feedback Loops and proposes a middleware between the independent engineering tools providing the required collaboration functionality.

2.2. Problem Domain of Systems Integration and Data Sharing

The problem domain of systems integration and data sharing includes the challenge of making it possible for heterogeneous software systems to work together seamlessly. The resulting data is then distributed to various sources and the engineering disciplines involved apply different and, thus, incompatible nomenclatures. Additionally, current integration approaches focus on describing technical details of their solution without assessing how to align incompatible nomenclatures. Finally, current integration solutions apply comprehensive integration technology with no consideration for the usability and acceptance of the solution by its future users, instead, current solutions require in-depth knowledge of systems integration. These major challenges in the problem domain of systems integration are further assessed in the following.

2.2.1. Heterogeneous Software Systems

Software tools applied during engineering design processes are mainly designed to complete a particular task for a single engineer rather than to integrate the processes along the automation systems life cycle (Biffl et al., 2009), and an integrated framework for the construction of mechatronic systems is lacking (Schafer and Wehrheim, 2007). A reason for this deficiency is that the discipline specific software tools fulfil individual requirements of each engineering discipline much better than an integrated software suite (Drath, 2010).
Integrated software suites often consist of a predefined set of tools and a homogeneous common data model, which work well for their intended purpose but do not easily extend to other tools in the project outside the tools' intended use (Waltersdorfer et al., 2010). Integrated software suites can only with difficulty be adapted to individual requirements or be extended by the integration of additional software tools. The high licence cost and hardware requirements of these integrated tool suites represent further problems for operators and producers of industrial manufacturing systems (Drath, 2010).

2.2.2. Distributed Data

In terms of systems integration in digital manufacturing, data is distributed among different software systems across the whole life cycle of a manufacturing system and a variety of CAE, PDM and VR software tools in use. Understanding Product Lifecycle Management (PLM) as the integration of CAE, PDM and VR applications, previous research and commercial PLM systems, offer only a limited interface with these applications (Song et al., 2009). A coherent PLM approach is still needed today and, thus, a seamless computer-aided planning of industrial manufacturing systems is also absent today (Drath, 2010). Most previous researchers have focused on the management and exchange of product information. However, PLM requires optimized data management through all stages of digital manufacturing (Chryssolouris et al., 2009) including the efficient management and exchange of product, process and resource information (Choi and Yoon, 2010).
Current standards in PLM can scarcely be applied due to the absence of a PLM system which supports the standards (Choi and Yoon, 2010). Moreover, the approaches of standardisation initiatives to formalising the knowledge across the life cycle of manufacturing systems is rather prescriptive, because it forces users to translate information from generic concepts to more practical and ad-hoc ones (Tursi et al., 2009).

2.2.3. Incompatible Nomenclatures

Product structure is a core discipline of systems integration, as it structurally connects the modules, items and data of a product (Brière-Côté et al., 2010, Schuh et al., 2008). However, well-structured product representation (Brandt et al., 2008) is lacking.
Understanding a manufacturing system as the product of the machine builder, information models representing this product structure of a manufacturing system are still lacking today (Brandt et al., 2008). Furthermore, the definition of modelling strategies for product structuring has only attracted scant attention from researchers (Reuter et al., 2010).
Today, the models representing a product structure in the engineering disciplines and their specific tools often use a range of terms and/or modelling structures to describe a particular concept, leading to semantically heterogeneous models (Moser and Biffl, 2010, Waltersdorfer et al., 2010). These terminological differences create misunderstandings and lead to various interpretations in the knowledge exchange process (Khilwani et al., 2009) making the lack of model integration between the en-

gineering disciplines due to different and incompatible nomenclatures one of the most conspicuous problems in current industrial development and research approaches (Waltersdorfer et al., 2010).
From a practical perspective, one reason it is difficult and time consuming to link semantically heterogeneous models is that it requires both domain and technical expertise: a domain expert who understands the domain meaning of all model elements being reconciled and a technical expert for writing transformations (Moser and Biffl, 2010). However, none of the current heterogeneous models is optimal for all engineers involved and engineers need to make a compromise when one model is adapted to another (Mascardi et al., 2010).

2.2.4. Lack of Introduction Strategies

An analysis of the existing literature on systems integration in mechatronic engineering design processes of manufacturing systems indicates that issues concerning the introduction of integration solutions to industrial environments have so far attracted little attention (Schuh et al., 2008). Existing approaches focus too much on general aspects or very low-level automation systems design and do not sufficiently address reconfigurability of manufacturing assembly systems in terms of their hard/physical and soft/logical aspects, as required for successful application in industry (Haq et al., 2010). Systems integration is further limited by the lack of tools and available techniques to assist in implementing the approach (Pullan et al., 2010). Thus, systems integration in industrial engineering design processes of manufacturing systems is still in the initial stages meaning that full and coherent systems integration is urgently required (Schuh et al., 2008).

2.2.5. Lack of User Acceptance

Business changes required for successful systems integration in all manufacturing sectors are not limited to technical systems; it is also essential to extend acceptance to the engineers, the users (Haq et al., 2010). However, products that result from high-end academic research often needed highly trained, highly scientific minds to operate them(Chryssolouris et al., 2009), which cannot be expected in industrial reality especially in small and mid-size companies. Accordingly, the users of integrated systems need to be intimately involved in its development process.

2.3. Systems Integration at the Conceptual Level

Systems integration approaches at the conceptual level include general considerations about enabling data and knowledge sharing in interdisciplinary engineering design processes.

2.3.1. Data Sharing

Since more than one software system is applied for engineering design, the need to exchange data and integrate the software systems has grown. One of the first approaches was the Integrated Computer-Aided Manufacturing (ICAM) program, funded by the U.S. Air Force in 1976 (McLean et al., 1983). The National Bureau of Standards built upon the results of this program and defined the Initial Graphics Exchange Specification (IGES) in January 1980 (Smith, 1983). This national standard led to the international Standard for the Exchange of Product Data (STEP) defined in ISO10303 initially released in 1994/95. In total, STEP consists of several hundred parts and every year new parts are added or new revisions of older parts are released making STEP the largest standard in ISO10303 (Anderl and Trippner, 2000).

The concept of Computer Integrated Manufacturing (CIM), which aims at centrally providing all manufacturing related data, was developed in the mid-1980s (Scheer, 2000). General rules on how to set up a CIM System were defined by the ESPIRIT consortium AMICE in the CIM Open System Architecture (CIMOSA) project between 1985 and 1995 (ESPRIT Consortium AMICE, 1993). The scope of CIM, which initially focused on manufacturing, has been widened to the whole product life cycle, and the field of Product Lifecycle Management (PLM) has also been introduced. A PLM system is used for the management of product data, which are usually 3D CAD data and product part lists which can be accessed worldwide through a network connection Siemens PLM Software (2013, *d*). A PLM system is a global solution used throughout the company, from the development division to the tools and equipment development division to the production and sales division. An enormous effort is necessary for the introduction and implementation of a PLM system.

2.3.2. Knowledge Sharing

Ontologies are increasingly exploited to share knowledge within and outside the boundaries of companies and other organizations (Mascardi et al., 2010). Neches et al. (1991) was one of the first researchers who transferred the term *ontology* from

philosophy to the field of artificial intelligence. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names are meant to denote, while formal axioms limit the interpretation and well-formed use of these terms (Gruber et al., 1993).

Neches et al. (1991) was motivated by the fact that knowledge base construction remains one of the major costs in building an artificial intelligence system: for almost every system built, a new knowledge base must be constructed from scratch. To overcome this barrier, we must find ways of preserving, sharing, reusing and building on existing knowledge bases. To archive this aim, Gruber et al. (1993) provided one of the first implementation concepts, called *Ontolingua*, a system for describing ontologies in a form that is compatible with multiple representation languages. It provides forms for defining classes, relations, functions, objects, as well as theories and translates definitions written in a standard, declarative language into the forms that can be input into a variety of representation systems.

Cutkosky et al. (1993) transfers the approach of Neches et al. (1991) and Gruber et al. (1993) to collaborative and concurrent engineering design processes. He was part of one of the research groups which jointly developed the Palo Alto Collaborative Testbed (PACT), a concurrent engineering infrastructure that encompasses multiple sites, subsystems and disciplines. One of the most recent research projects in the field of knowledge sharing in collaborative and interdisciplinary engineering design processes is the Engineering Knowledge Base (EKB) introduced by Moser et al. (2010). It is a semantic-web-based framework, which supports the efficient integration of information originating from different expert domains without a complete common data schema.

2.4. Systems Integration at the Physical Level

In contrast to the conceptual level, systems integration approaches at the physical level are more concerned about the accompanying implementation issues. This includes specific ways of translating data from one data model to another as well as transferring data between various software systems.

2.4.1. Data Translation

The aim of research activities in the field of data translation is to reduce the time it takes human experts to create a mapping between a pair of data schemas. To fully support interoperability, the federated systems must share a set of commonly understood concepts and relationships (Mascardi et al., 2010). Ontologies are a promising concept to enable the semi-automated data transfer between data schemas (Moser and Bifflˆ, 2010). Stuckenschmidt and Van Harmelen (2005) distinguish between three different ways of employing ontologies for data translation: single ontology approaches, multiple ontologies approaches and hybrid approaches (see Figure 2.1). Single ontology approaches use one global ontology to provide a shared vocabulary for the specification of the semantics (see Figure 2.1a). All information sources are related to one global ontology. In multiple ontology approaches, each information source is described by its own ontology (see Figure 2.1b). Similarly to multiple ontology approaches, in hybrid approaches the semantics of each source is described

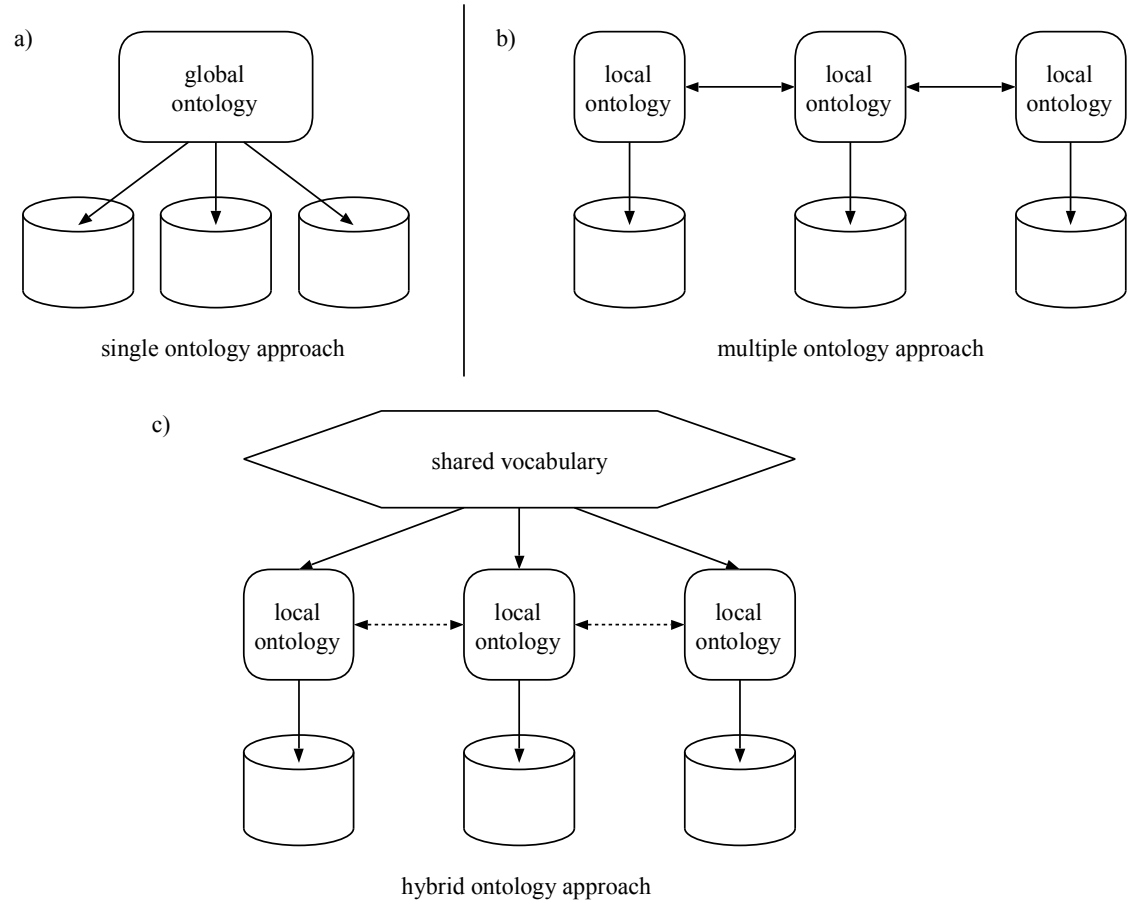


Figure 2.1.: The three different ways of employing ontologies for data translation (Stuckenschmidt and Van Harmelen, 2005).

by its own ontology (see Figure 2.1c). However, in order to make the source ontologies comparable to each other they are constructed upon one global shared vocabulary. The advantage of the hybrid approach is that new sources can be easily added without modifying the mappings or the shared vocabulary.
Based on Stuckenschmidt and Van Harmelen (2005), Zhang et al. (2009) proposes a multiple ontology approach for multidisciplinary design application, which consists of a core ontology, a discipline ontology and a cross-disciplinary ontology. Developing an ontology-based multidisciplinary modelling framework is a step towards a general ontology of multidisciplinary design, which is needed to support semantic annotation for access and retrieval of multidisciplinary design knowledge in the distributed, collaborative design environment.

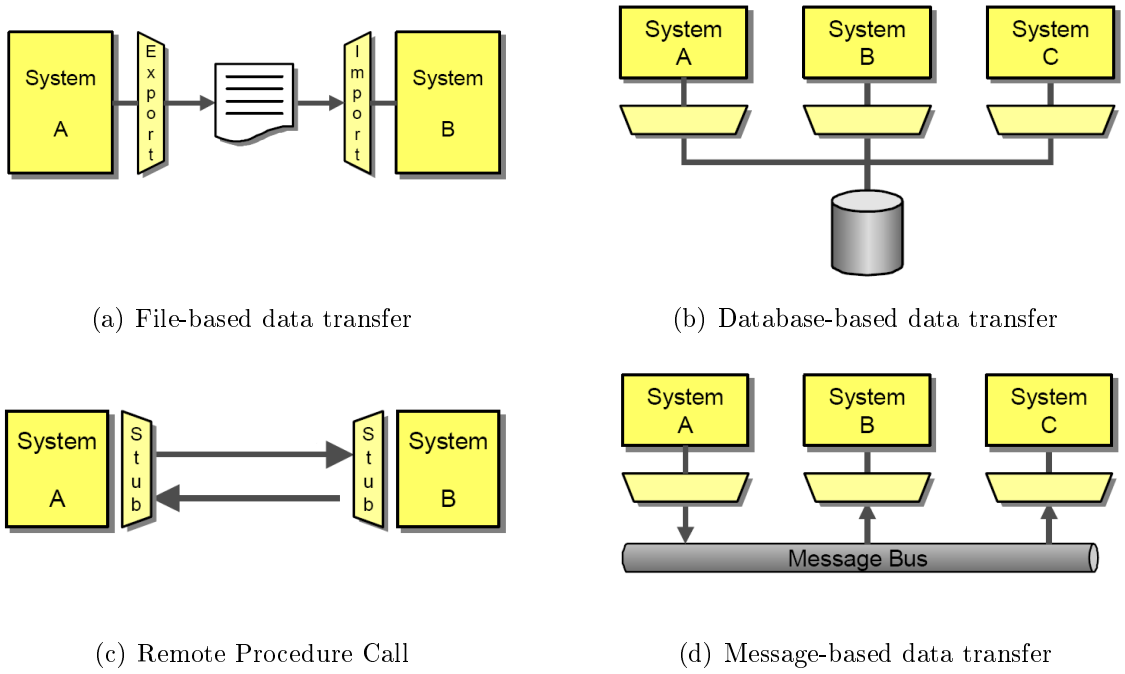
Despite the many component/ontology matching solutions that have been developed so far, no integrated solution is a clear success, robust enough to be the basis for future development, and usable by non expert users (Shvaiko and Euzenat, 2008). Consequently, much more work needs to be done in order to bring the ontology matching technology to the plateau of productivity. A more pragmatic approach which is currently being pursued by industrial companies, is the application of open file format, which reduces the number of converters to the number of applied software tools. Ideally, each software tool needs to support the open format in addition to its native format (Drath, 2010). However, for the definition of such an open file format it is also necessary to define common concepts and to find a common language among the stakeholders using it. In this way, open file formats are comparable to ontology definition.

2.4.2. Data Transfer

According to Hohpe and Woolf (2004), a common classification system for data transfer solutions is a distinction according to their implementation methods, whether it is based on file exchange, a central database, remote procedure calls or message exchange. A scheme of these principle implementation possibilities is illustrated in Figure 2.2.

2.4.2.1. File-based data transfer

In its simplest form, data transfer can be performed by file exchange. A promising approach addressing interdisciplinary data exchange during the engineering design process of manufacturing systems is the Automation Modelling Language



(a) File-based data transfer

(b) Database-based data transfer

(c) Remote Procedure Call

(d) Message-based data transfer

Figure 2.2.: Classification of data transfer solutions by their implementation according to Hohpe (2002)

(AutomationML) (Drath et al., 2008, Drath and Miegel, 2009). This approach is a combination of three different exchange formats: the PLCopen XML format (PLCopen Technical Committee 6, 2009), for exchanging programs for PLCs; the Computer Aided Engineering eXchange (CAEX) format (Schleipen et al., 2008), for exchanging hierarchical object information of manufacturing systems; and the COLLADA format for exchanging 3D CAD models and kinematics (Arnaud and Barnes, 2006).

These formats are all XML-based file formats (Bray et al., 2008) which have the advantage that they can be easily processed because the majority of programming languages provide support for serialisation of object models to XML. Furthermore, XML-based files can be accessed by simple queries comparable to querying a database.

However, a disadvantage of file based integration approaches is the requirement to define a protocol to be used by engineers on when to read and/or write to file (Hohpe and Woolf, 2004). The file update (or synchronisation) is usually carried out at regular intervals (monthly, weekly, daily,...) to ensure consistency and reflect the most recent changes. When an update occurs earlier than expected, the affected systems can lose synchronisation. Furthermore, users have to agree on file-naming conventions and locations which is often a source of resistance. Moreover, file names must be unique and users have to agree on the conditions when obsolete files can be

deleted. When the complexity of designs increases, the quantity and complexity of the exchanged data increases and this results in agreements which are more difficult to manage.

2.4.2.2. Database-based data transfer

Due to the limitations of file-based data transfer, other approaches have adopted a centralised database strategy such as Computer Integrated Manufacturing (Scheer, 1990). While the scope of CIM was limited to manufacturing processes, modern integration approaches have a wider view on the whole product life cycle process. These approaches are usually termed Product Lifecycle Management (PLM) (Siemens PLM Software, 2013, *a*). Waltersdorfer et al. (2010) introduced the Engineering Database (EDB) concept, which provides the foundations for version management and update conflict detection in engineering data models across tool boundaries.
Database based solutions make data availability more efficient and enforce an agreement through centralised storage of data. However, database solutions have their disadvantages, too. A shared database results in a large data structure which makes it much more difficult to encapsulate and hide information. Un-encapsulated data in a shared database makes it difficult to preserve the integrity in case of changes of the integrated software systems (Hohpe and Woolf, 2004).

2.4.2.3. Remote procedure calls

Remote procedure calls allow an application to invoke a function implemented in another application. They provide standard access to data repositories without the need for building a different interface for each application (Bakis et al., 2007). Examples for remote procedure call standards are CORBA, COM+, JavaBeans or .Net. They have been developed to support the frequent exchange of small amounts of data. This downgrades the performance in engineering design environments where a more asynchronous type of communication and the exchange of large volumes of data are usually required (Bakis et al., 2007). Synchronous point to point connections between senders and receivers established by remote procedure calls become difficult to maintain as the number of participants increases (Hohpe, 2002).
In the field of Computer Aided Engineering (CAE) Song et al. (2009) presents the CAE2VR middleware for CAE data exchange. It allows the calling of functions for the creation of geometric elements as well as the evaluation and visualisation of analysis data in remote software tools.

2.4.2.4. Message-based data transfer

Message-base data transfer is data transfer based on the exchange of messages with the aid of a Message-Oriented Middleware (MOM). There is no general standard for the content of a message. However, XML-based messages are usually exchanged. As long as the systems which are supposed to exchange data with the help of messages support the message oriented middleware they can exchange data across (operating) system boundaries. Additionally, encapsulation of all information exchanged in a clearly defined message allows for the independent development of the software tools involved in the integrated system. A disadvantage, though, is that if the MOM stops running, the whole system will stop running.
The Java Message System is a common MOM and there are several commercial (e.g. IBM Websphere MQ) and open source (e.g. Apache Active MQ) JMS providers available.
A MOM forms the backbone of an Enterprise Service bus. It is a standards-based integration platform that combines messaging, web services, data transformation and intelligent routing to reliably connect and coordinate the interaction of significant numbers of diverse applications (Chappell, 2004). The Automation Service Bus approach applies proven concepts from the Enterprise Service Bus in the business IT context (Hohpe and Woolf, 2004) to automation systems engineering (Biffl et al., 2009).

2.5. Conclusions and Discussion

The problem domain of systems integration and data sharing includes 5 major fields: heterogeneous software systems, distributed data, incompatible nomenclatures and a lack of both integration strategies and user acceptance.
It must be assumed that future trends will not improve the fields of heterogeneous software systems and distributed data. Due to the increasing specialisation of the engineering disciplines and the fact that the number of disciplines is increasing (e.g. growing importance of IT and diagnosis), the number of heterogeneous software tools will not decrease. Furthermore, specialised software tools will maintain their advantages in comparison with integrated tool suites due to their ability to react to changing user requirements with greater agility. Thus, data will remain distributed which is also enforced by the trend towards more globalised and distributed projects leading to more globalised and distributed engineering teams.
As such, homogeneous software tools and distributed data can be seen as surrounding conditions which can hardly be influenced to improve the process.

Therefore, activities should focus on aligning and standardising the nomenclatures and improving integration strategies as well as user acceptance of the systems integration and data sharing solutions.

The problem of incompatible nomenclatures arose from the fact that the engineering disciplines concerned have a similar yet different understanding of the same physical concepts. As in many other fields, the different nomenclatures for similar concepts arise from the historical development of the various engineering disciplines, which were more independent of each other in the past. It is quite common for engineering disciplines to have individual differentiations and groupings of terms as well as different levels of detail. It is very difficult to align the nomenclatures because it requires compromises and changes in habits.

The current approaches to systems integration and data exchange insufficiently consider how they can be introduced to an industrial engineering design process. Most of these approaches cannot be integrated stepwise but need to be introduced as a whole in a big-bang integration. Such an integration strategy is connected with a huge technical, organisational and thus financial risk. Thus, methods for a stepwise introduction of an integration solution are required.

Problems in regard to the lack of user acceptance of present approaches are caused by high requirements on the qualifications of the engineers involved. Moreover, the engineers need to support the changes to their familiar software landscape and business processes. The user acceptance might also dwindle due to user concerns that they will become redundant due to increasing automation and growing efficiency.

3. Analysis of Systems Integration in Mechatronic Engineering Design Processes

The literature review in the previous chapter identified incompatible nomenclatures as one of the major problems of systems integration. A domain specific nomenclature is reflected in the data models underlying the specific software tools applied (Conrad et al., 2006). Hence, from a technical point of view, aligning nomenclatures means to align data models applying a certain modelling strategy which is why assessing systems integration from a modelling point of view is the key motivation of this research.
This chapter demonstrates the relevance of data modelling for systems integration by referring to data models from the engineering design process of body shop production lines. Current top-down and bottom-up data modelling strategies are assessed and compared with respect to their potential for dealing with the other major fields in the problem domain of systems integration, among which are user acceptance and introduction strategies.

3.1. Relevance of Data Modelling for Systems Integration

The relevance of data modelling for systems integration can be demonstrated with an example from the mechatronic engineering design process of body shop production lines. This example confirms the results of Moser and Bifflˆ (2010) by showing that different mechatronics engineering disciplines have different perspectives on the same physical objects or concepts, which in this case is a jig. These disciplines apply their domain specific nomenclatures and, thus, have different data models in their specific software tools. Transferring data between these data models gives rise to a number of challenges which are also assessed in the following.

That mechanical and electrical engineers have different perspectives on the same physical design object is demonstrated by reference to a jig as a typical mechatronic component of a manufacturing system.
In a body shop production line a jig holds two or more car body sheets in position so that a robot can join (e.g. weld, clinch, glue, etc.) them (Fenton, 1998). A jig has to be meticulously designed to cover both mechanical and electrical aspects. Engineers use their specific engineering design software systems and store their design information in different data models as illustrated in Figure 3.1.

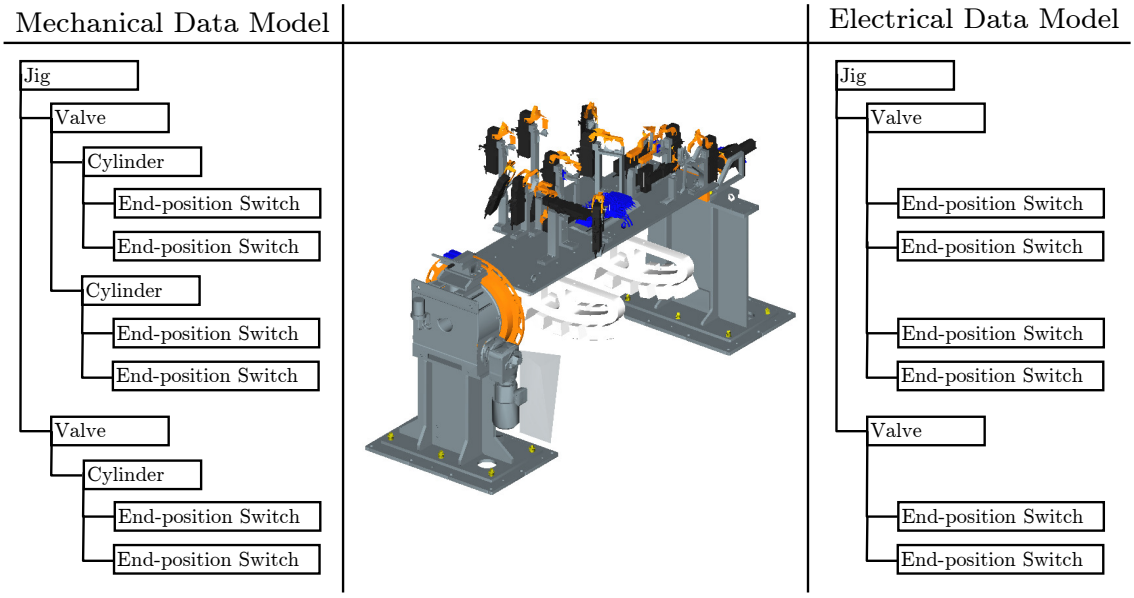


Figure 3.1.: Different data models of a jig

The data models of a jig, the mechanical and the electrical are similar but not equal due to the specific requirements of each engineering discipline: Sub-components which shape the form of a jig are relevant for its mechanical design. Thus, the mechanical data model contains complex data types of all sub-components with a spatial characteristic: valve, end-position switch and cylinder.
In contrast to this, the electrical data model only contains the valves and end-position switches because only these sub-components are required for the electrical control of the jig.

In transferring data from the mechanical data model to the electrical, with the exception of the cylinder, complex data types are not transferred because they are not relevant for the electrical design. Data transfer in the other direction, from the electrical data model to the mechanical is more complex because the information stored

in the complex data type of the cylinder does not exist. It is possible that the electrical engineer has to make some changes, such as choosing a different end-position switch which requires a change in the design of the cylinder. Then the danger of losing consistency in the data model due to contradictory information relating to a cylinder with a different end-position switch can arise. Manual intervention is required to check whether the changed end-position switch works with the existing design of the cylinder, since it does not exist in the electrical data model. This check could be automated when all combinations of possible end-position switches and cylinders are formally described in a transfer algorithm. Such a transfer algorithm is maintenance-intensive because it has to be adapted as soon as any changes occur.
In summary, the design of the data model is dependent on the specific requirements of an engineering discipline. Due to the fact that these requirements differ from discipline to discipline, the data models in the software systems differ as well. Maintaining consistency requires either manual interventions or in the case of an automated process, it is necessary to implement appropriate transfer algorithms. However, these transfer algorithms need to be continuously modified to keep up with the changes in the data models resulting from new/enhanced designs.

3.2. Classification of Current Data Modelling Strategies

For the classification of current data modelling strategies, this research assesses data modelling at two levels, the tool level and the integration level. Each of these levels has certain constraints and requirements on a particular data model.
At the tool level the data model is designed with respect to the functionality of the software tool. It reflects the domain specific understanding of an engineering discipline. Since functionality differs from software tool to software tool just as the domain specific understanding differs from discipline to discipline, the applied data models also differ.
Tree diagrams in different shapes and colours are used to illustrate the difference between data models at the tool level. They represent data models underlying the three different software tools applied, for example, during mechanical engineering, electrical engineering and software development.

Advanced systems integration approaches require a common data model at the integration level. This common data model includes all information processed by the

involved engineering disciplines during their design work (Conrad et al., 2006).
This research focuses on strategies for establishing a common data model for the integration level to enable advanced systems integration. Current approaches establish such data models either top-down, from the data exchange level down to the tool level, or bottom-up, from the tool level up to the data exchange level.

3.2.1. Top-Down Data Modelling Strategy

The underlying concept of the top-down modelling strategy involves establishing a generic data model independently of the data models at the tool level in order to be suitable for a wide range of integration scenarios. Accordingly, in the majority of cases there is only a minor correspondence between the data model at the integration level and the data model at the tool level. The top-down data modelling strategy is more focused on assuring the consistency of the information stored by avoiding redundant storage of the same information in different parts or elements of the data model.
However, such a generalised data model may not contain all the required information or contain too much information. In the former case, an extension of the generalised data model is required to facilitate the data transfer, while in the latter case the extension would increase the processing overhead unnecessarily. Unfortunately, users and software producers have only limited influence on the generalised data format since the development is usually controlled by a standardisation body.

The general approach of the top-down data modelling strategy is illustrated in Figure 3.2. The data model with the grey elements on the integration level represents the top-down data model. Grey elements with coloured squares in the middle represent corresponding information between tool specific data models and the top-down data model. As described above, these links are not 1:1 but n:m.

The data model supporting the AutomationML offers a good example of an integration approach adopting a top-down data modelling strategy. AutomationML consists of a set of general objects (e.g. Object, Group, Class, Attribute, ...) and a number of general techniques to connect and further describe these objects (e.g. Interface, Port, Facet, ...) (Automation ML consortium, 2013). It does not include setting, interpreting or exchanging company specific standards. Such details are left to the software tools and for the companies involved in the data exchange to resolve (Drath, 2010). Therefore, general concepts in top-down designed data models need to be adapted to address the more practical and unexpected issues that are typical

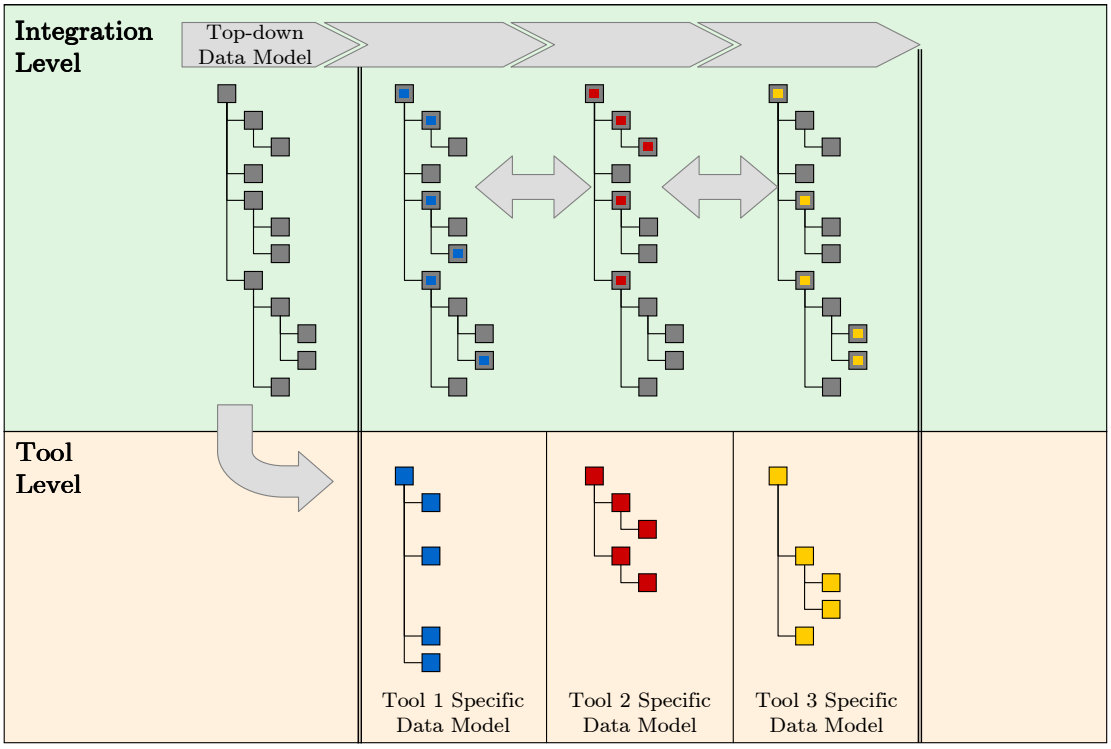


Figure 3.2.: Top-down data modelling strategy

in industrial engineering processes (Tursi et al., 2009).

3.2.2. Bottom-Up Data Modelling Strategy

A bottom-up data modelling approach is an unplanned data modelling strategy and can be observed in many mechatronic engineering design processes in industrial practice. It takes as a starting point present data models at the tool level. The common data model for the integration level is created by adding together all data models at the tool level and allows a simple and stepwise integration of one software tool after another. This bottom-up data modelling approach has been observed in industrial engineering processes by a number of researchers, for example Bergert et al. (2007), Qi et al. (2006).

Nearly every software tool offers simple mechanisms to support rudimentary data exchange. These mechanisms generally provide import and export interfaces to enable data exchange via text-based file formats or a user interface to allow copy and paste content from the clipboard. In addition to some fundamental agreements between the engineering disciplines concerned such as naming conventions, these

simple data exchange mechanisms have already proved their value in improving the engineering design process. In contrast to the top-down strategy, the bottom-up strategy has delivered data exchange possibilities without reference to global concepts, which is why they are referred to as bottom-up data modelling strategy in contrast to the top-down data modelling strategy which is based upon global concepts in the particular process.

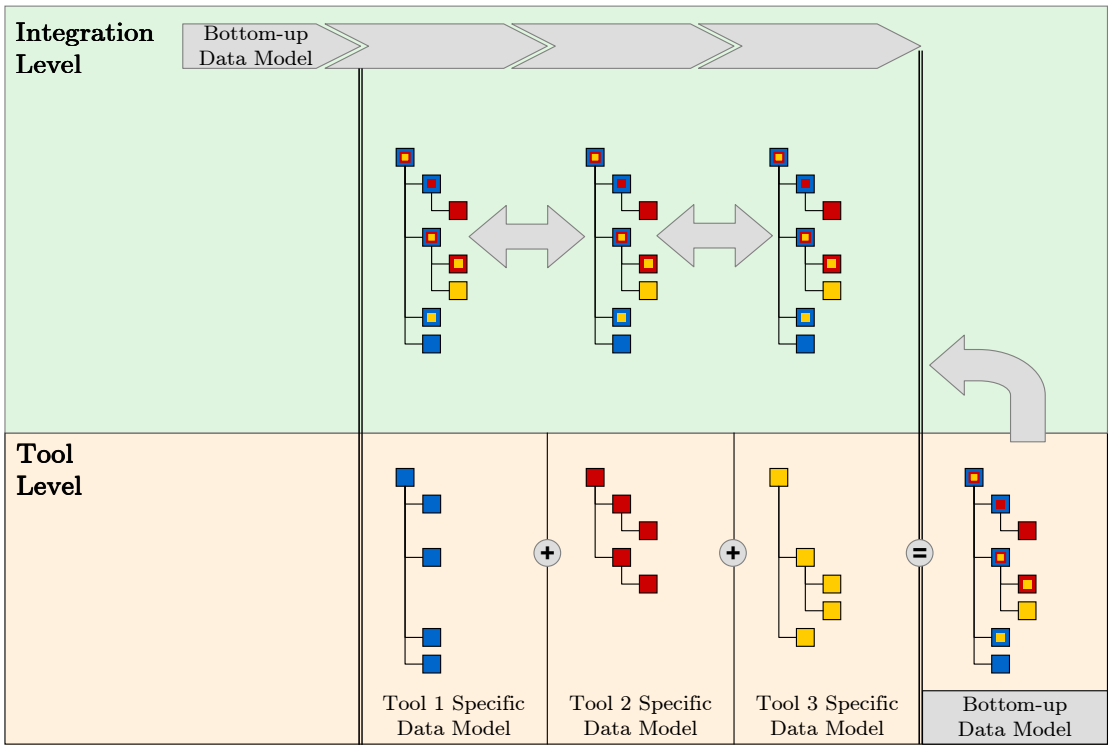


Figure 3.3.: Bottom-up data modelling strategy

Figure 3.3 shows the principle approach of the bottom-up strategy. As in Figure 3.2 three different data models at the tool level are shown. The bottom-up data model is created by adding up the tool specific data models. It is possible to track the source data model in the common data model with the colour of the elements. Parts with only one colour, either blue, red or yellow, only appear in one data model at the tool level. Elements with two or three colours indicate overlapping parts of two or three data models at the tool level.

The overlapping parts contain information on physical objects or concepts as they are required for the design work in a number of engineering disciplines. However, the information is structured differently from discipline to discipline, according to its domain specific understanding. To integrate this information in one common data model, the engineering disciplines involved need to agree on how to commonly store

them. However, global concepts for the design of the data model at the integration level are lacking. In many cases, the resulting elaborate unification process leads to unfavourable data models at the integration level. Similar information is likely to be stored in several parts of the data model making it difficult to maintain consistency when changes occur that only consider aligning one part.

3.3. Shortcomings of Current Data Modelling Strategies

In this section the ability of current data modelling strategies to meet the challenges of data modelling in the top-down and bottom-up strategies is discussed, compared and contrasted. In this way shortcomings of these two current approaches are exposed. While the advantage of the top-down approach over the bottom-up approach lays in maintaining the data model consistency, the advantage of the bottom-up approach is in fulfilling multidisciplinary requirements. Additional results of the comparison are summarised in Table 3.1.

	Bottom-up	**Top-down**
Maintenance of data model consistency	Effects on the data model caused by changes in a certain part are hard to survey −	Data model consistency is considered during its design +
Fulfilment of multidisciplinary requirements	Allows stepwise consideration of each discipline's requirements +	Requirements which are not considered during early modelling can hardly be incorporated later on −
Introduction strategy to industrial processes	Presently applied software tools can be utilised further on +	Presently applied software tools need to be adapted or exchanged −

Table 3.1.: Strengths and shortcomings of current data modelling strategies

The consistency of data models created using a top-down modelling approach is usually ensured because all information, links, dependencies and constraints have already been anticipated during the design stage. Thus, there should be no contradictory information stored. However, specific requirements of individual disciplines which were not considered during the early top-down design phase, cannot be readily incorporated later. Moreover, difficulties might arise in fulfilling

specific requirements of an individual discipline in order to maintain the consistency of the data model.
In order to introduce a top-down data model to a current engineering design process, either the underlying data structure of a tool is changed or transfer algorithms must be added to the software tool to translate the data from the integration to the tool level. Changing the underlying data model of a software tool often necessitates a complete redesign, which is a substantial effort. Transfer algorithms need to be continuously maintained when changes in the data models at the tool or integration level occur.

Data modelling with a bottom-up strategy makes it possible to consider the requirements of each discipline step-by-step since it is possible to integrate one discipline-specific data-model after another into the data model. However, integrating numerous data-models rapidly increases the complexity of a data model, hence making it difficult to manage. For example, gaining an overview of the links, dependencies and constraints between the newly integrated data and the rest of the data model can be quite difficult, which makes it more complicated to navigate during a review. Thus, data model consistency is difficult to maintain when approaching data modelling with the bottom-up strategy.

3.4. Summary

Current systems integration and data exchange approaches follow either a top-down or a bottom-up data modelling strategy. These strategies differ in their capacities for handling major challenges in data modelling. A bottom-up data model grows in complexity the more systems are integrated; thus, its consistency is more difficult to preserve than a top-down data model, where consistency is considered from the start of modelling. However, the bottom-up strategy has the advantage that it requires fewer changes in presently applied software tools. Additionally, systems integration based on a bottom-up data modelling strategy can be introduced to industrial engineering processes in steps.
Since data modelling is identified as a key enabling factor for successful data exchange and systems integration, a new data modelling strategy is required which combines the advantages of current top-down and bottom-up approaches. It should allow the design and maintenance of a consistent data model which can be introduced in steps to an industrial engineering process with minimal changes in the applied software tools.

4. New Approach To Systems Integration

This chapter presents a new approach to data exchange in interdisciplinary engineering design processes based on a new middle-in data modelling strategy. It combines the advantages of the bottom-up and the top-down strategies. It enables a consistent data model to be established which fulfils the requirements of all engineering disciplines involved in the engineering design process.
A method for determining the core data model and detailed design data modules as the two major elements of a middle-in data modelling strategy is illustrated. At first this is undertaken specifically for data occurring during the engineering design process of body shop production lines. After this, insights and experience gained from the specific application are used to formalise a general description of the middle-in data modelling strategy by means of set theory.

4.1. New Middle-in Data Modelling Strategy

The new middle-in data modelling strategy is distinguished from the bottom-up and the top-down data modelling strategies by assessing them on the levels of data modelling, tool level and integration level, as introduced in the previous chapter. The middle-in modelling strategy includes the division of the data model at the integration level into two parts. One part is created by applying the top-down data modelling strategy and other part is created by applying the bottom-up data modelling strategy, which is why this approach is referred to as the middle-in data modelling strategy.

The part which is created by employing the top-down approach is called the *Core Data Model*. It contains the information relevant for all engineering disciplines involved in the data exchange process. Each engineering discipline is responsible for maintaining the consistency in this part of the data model. The parts of the data model which are created bottom-up are called *(Detailed Design) Data Modules*.

One data module contains the information relevant for only a few (typically two) of the engineering disciplines involved. The engineering discipline producing the information contained in a data module is responsible for preserving data consistency with the core data model.

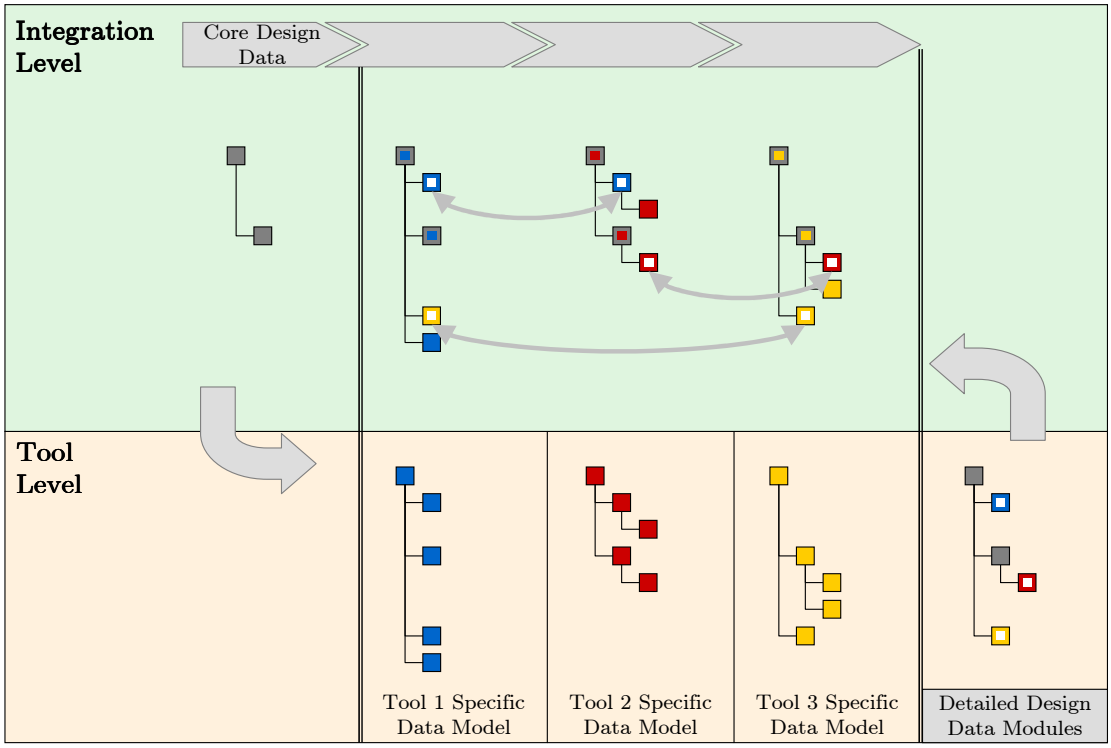


Figure 4.1.: Middle-in data modelling strategy

Figure 4.1 illustrates the middle-in data modelling strategy. As in the illustrations of the other strategies, this figure includes the three different data models at the tool level. The data model at the integration level also includes two grey elements, which belong to the core design data because they can be found in all three data models at the tool level.

The detailed design data modules in the data model of the integration level are illustrated as blue, red and yellow elements with a white box in the middle. They include design information which goes beyond the core design data and is not relevant to all engineering disciplines involved. In the example in Figure 4.1, the content of the detailed design data modules are relevant to two disciplines.

4.2. Middle-in Design of Engineering Data

The middle-in data modelling strategy is applied to the mechatronic engineering design process of body shop production lines. Present data structures from the different mechatronic disciplines are analysed with regard to similarities and differences. The similarities of present data structures are aligned and standardised in order to define the core data model. For data not contained in the core design data, an example of a detailed design data module is provided.

4.3. Comparison of Current Data Models

The engineering process of body shop production lines includes a variety of software tools which support mechatronic design. For the identification of the core design data of a body shop production line, the data models underlying two of the major software tools are compared with respect to the hierarchical structure of the information and the applied nomenclature for elements and concepts. The two software tools considered are *ProcessDesigner* (Siemens PLM Software, 2013, *b*) and *Eplan Engineering Center (EEC)* (EPLAN Software & Service, 2013, *b*). The ProcessDesigner is used for 3D layout design as well as the calculation of investment and commissioning costs, while EEC is employed for the automatic generation of electrical schematics and PLC programs.

Both software tools are set up on data models representing a complete body shop production line in its different hierarchical levels, illustrated as a tree view for both tools. The tree views of the same body shop production line developed from both software tools are compared with respect to similarities and differences in extracting the core design data of a body shop production line.

A tree view provides a useful visualisation of data structures for comparing the hierarchical structures and the nomenclature of their elements. Corresponding elements can be juxtaposed as was carried out for the data structures of the ProcessDesigner (left) and the EEC (right) in Figures 4.2 and 4.3 where corresponding elements are marked with a red line. The data models are compared on several levels. While Figure 4.2 shows the sub-components of a PLC area down to the operation level, Figure 4.3 shows the elements below the operation level. In order to keep the figure comprehensible, only the third safety circuit assigned to the PLC area is shown.

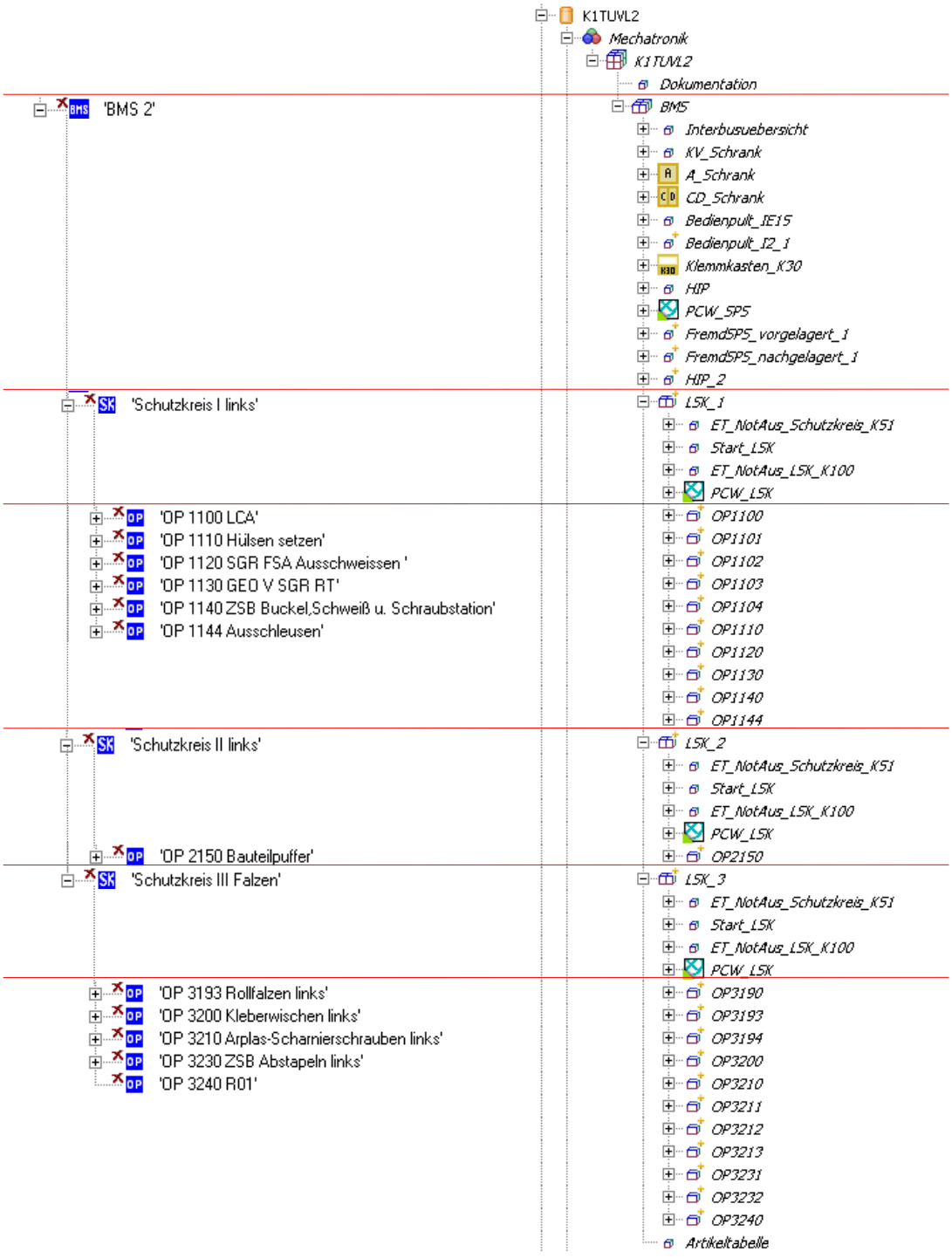


Figure 4.2.: Comparison of ProcessDesigner (left) and EEC (right) structures of a PLC Area (For a translation see Figure A.1 in the appendix)

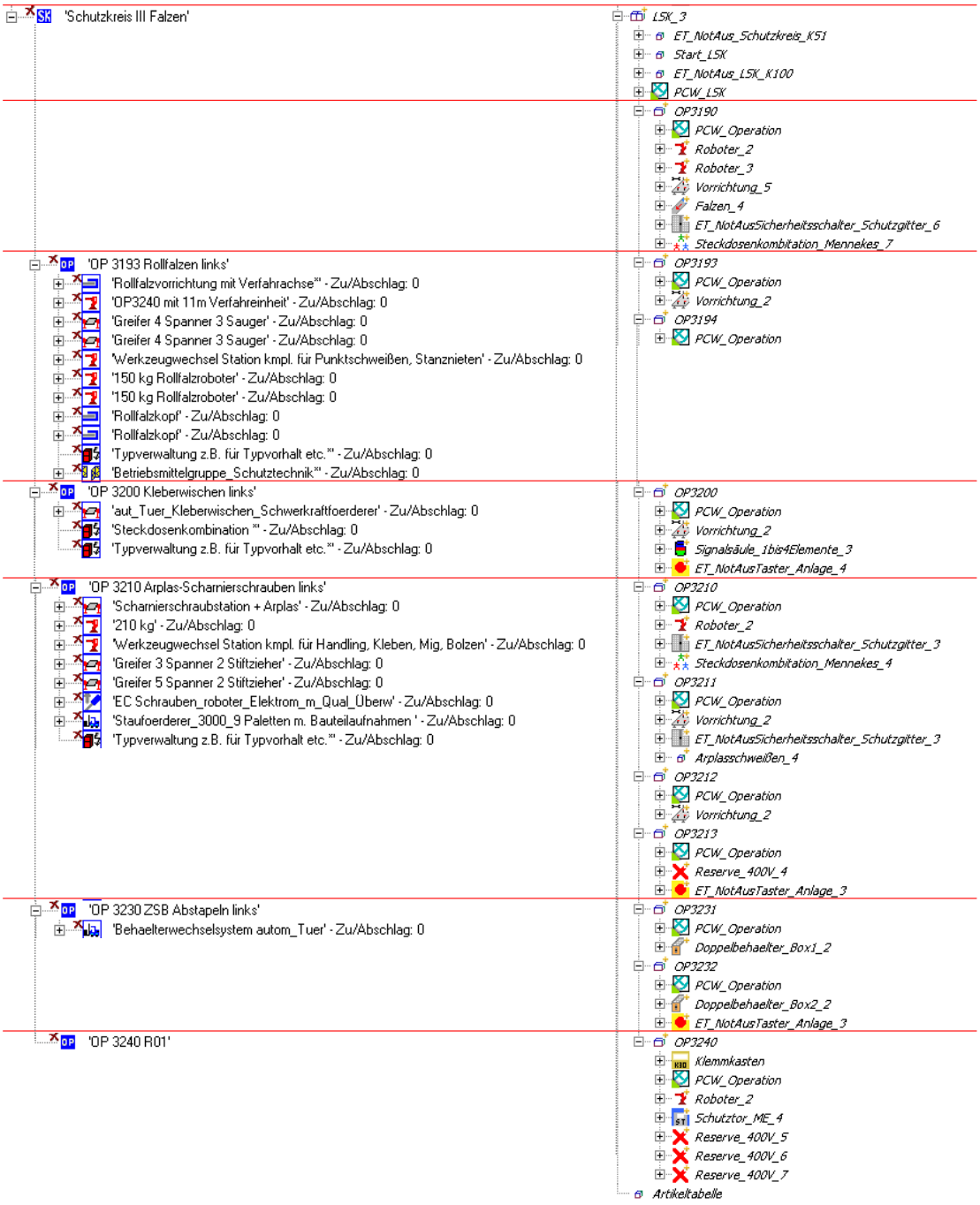


Figure 4.3.: Comparison of ProcessDesigner (left) and EEC (right) structures of a Safety Circuit (For a translation see Figure A.2 in the appendix)

The corresponding elements in this example are *PLC Area* (German: Betriebsmittelsteuerung - BMS), *Safety Circuit* (German: Schutzkreis - SK / Lastspannungskreis - LSK) and *Operation* (OP). However, these elements differ in their nomenclature as well as their level of detail.
A difference in nomenclature, for example, is that PLC areas in the EEC structure are not numbered because an EEC project can only contain one PLC area whereas a ProcessDesigner project can contain more than one. In addition, the name of a safety circuit is spelled in full and given a Roman numeral in the ProcessDesigner, while in the EEC it is abbreviated and assigned an Arabic numeral. Another difference is that while operations are spelled the same in ProcessDesigner and EEC, ProcessDesigner appends a short description of the functionality of an operation.

Below the operation level, corresponding elements cannot be directly matched (see Figure 4.3). At this level the division of corresponding elements differs between mechanical and electrical design. Another difference is the *Container Changing Station* (German: Behaelterwechselsystem) OP3230 shown in Figure 4.3: for mechanical design a container changing station is one piece of the design with one specific function. Accordingly, it has one dedicated operation number. For electrical design it is two boxes with identical equipment for mounting a container. In the EEC data structure, the container changing station consists of two operations: OP3231 and OP3232.
Besides these differences in hierarchical relations and the division of similar elements, a matching of corresponding elements is also hampered by different nomenclatures for similar elements. Jigs are called “GEO” (= geometry jig) or “Ablage” (= rest table) in the ProcessDesigner, while they are called “Vorrichtung” (= jig) in the EEC. As soon as more than one jig is contained in an operation, a one-to-one matching is not possible. The same can be said of robots: in ProcessDesigner they are designated by the weight they can carry (e.g. “150 kg”) or their name is a compound of the operation’s name + “R” + number of the Robot (e.g. “OP3210R01”). In EEC they are simply called “Roboter” (= robot).

In summary, the comparison of the data models from ProcessDesigner and EEC show that they are similar but not equal. The elements at the highest hierarchical levels correspond to each other, whereas the elements at the lowest hierarchical levels differ. However, even the corresponding elements differ in their nomenclature. To include them in the core design data, global concepts for aligning them must be introduced. These examples confirm the circumstance that different engineering

disciplines have different perspectives on the same physical objects which is impeding systems integration and data exchange as shown previously in Chapter 3.1.

4.3.1. Core Design Data of a Body Shop Production Line

The comparison of the data models underlying ProcessDesigner and EEC leads to a set of corresponding elements with corresponding hierarchical relations(as shown in the previous subsection). These elements, the *PLC Area*, the *Safety Circuit* and the *Operation*, as shown in Figure 4.4, reflect the basic functional structure of a body shop production line and its inherent task: connecting car body parts.

Dividing manufacturing systems into functional, mechatronic units and applying hierarchies of these units as the basis for data model design is also described by Mangold et al. (2003), Lüder et al. (2010) and Reuter et al. (2010). Their research includes analyses of manufacturing systems other than body shop production lines which underscores the broad applicability of dividing manufacturing systems into hierarchies of functional units. The functional hierarchy of a body shop production line as the basis for the core design data which is shared among the engineering disciplines involved is presented in the following.

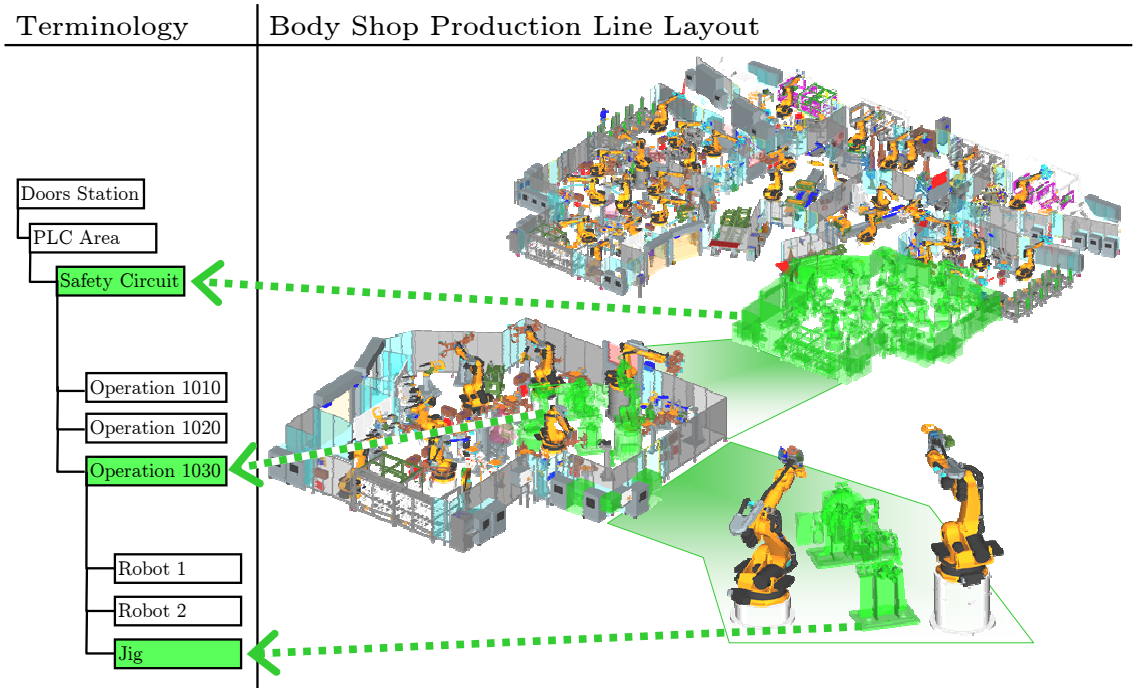


Figure 4.4.: Core data model of a body shop production line shared by all engineering disciplines

The task of the smallest functional unit in a body shop production line is the creation of one junction between two body sheets using a joining technology, for example a welding spot. Each body shop production line has a defined cycle time depending on the required number of output parts. Hence, several joining points are usually set during one production cycle. Normally during one cycle the body sheets remain in one jig where the junctures are set. After one cycle the joined body sheets are transported to the next jig or buffer. Thus, the systems used to join two body sheets are the lowest functional level, for example jigs or robots.
The next functional level, called *Operation*, consists of all systems used during one cycle and is numbered according to the production sequence. The subsequent functional level is defined with respect to the availability of the production line. In order to keep availability high, several operations are combined into *Safety Circuits*. In one safety circuit all components are safety-related with each other. When one component stops due to an error or a safety stop button is pressed, all other components in the safety circuit immediately stop as well, while the components from other safety circuits continue to run.
Safety circuits are decoupled by body sheet buffers so that further body sheets can still be produced, even when parts of the line are not working. The next functional level is the *PLC Area*, which comprises all safety circuits controlled by one PLC.

All engineering disciplines involved in the engineering design process share the same functional understanding of a body shop production line. This basic functional structure forms the core design data of a body shop production line as shown above. The information stored in the core design data can be seamlessly exchanged among the engineering disciplines involved because it is stored in every data model underlying the software tools.

4.3.2. Example of a Detailed Design Data Module

The core data model as presented above does not contain all the information produced and required by all engineering disciplines concerned. Additional data is stored in detailed design data modules. While data contained in the core data model is relevant to all engineering disciplines involved, the information stored in a detailed design data module is only relevant to a few disciplines.

The following example illustrates the detailed design data which is exchanged between 3D detail design and pneumatic schematics design in the engineering design process of body shop production lines. In 3D detail design, CAD models

of body-shop elements, such as jigs or grippers are designed in detail. Pneumatic Schematics Design specifies how the pneumatic components on a body-shop production line are supplied with compressed air.

Pneumatic schematics design builds upon information from the 3D detailed design regarding the types of cylinders and valves and the final position switches applied. Depending on the sequence of the body-shop production line, information regarding cylinder grouping and their sequence of action is added. However, for pneumatic schematics design no full 3D CAD drawing is required, only a carefully considered data set, which can be stored in a simple table. An extract of this information is shown in Table 4.1.

Cylinder Group 1	
Number of Cylinders	5

	Label	**Valve Type**	**Electric**	**Valve Properties**
Valve	KYP11	5-port/2-way	bistable	vended

	Label	**Cylinder Type**	**Basic Positon**	**Final Position Switch Type**
Cylinder 1	46-38D 821893	double-acting	retracted	switch set
Cylinder 2	46-38D 821891	double-acting	retracted	switch set
Cylinder 3	46-38D 821897	double-acting	retracted	switch set
Cylinder 4	46-38D 821907	double-acting	retracted	switch set
Cylinder 5	46-38D 821895	double-acting	retracted	switch set

Table 4.1.: Extract of mechanical detailed design data required for pneumatic schematics design

In order to clearly assign the detailed design information to the jig of a body-shop production line, it is linked to the complex component *Jig* in the core data model. Thus, which Operation, Safety Circuit, PLC Area and Sub Assembly the jig is assigned to can be clearly indicated.

4.4. General Approach to Middle-in Data Modelling

In addition to the specific application of the middle-in data modelling strategy to the mechatronic engineering design process of body shop production lines, a general approach to middle-in data modelling is derived which is introduced in this section, which is described by means of set theory, interpreting engineering data as

overlapping sets.The motivation for this general description of the middle-in data modelling strategy is to demonstrate its broader application beyond the engineering design process of body shop production lines.

Seeing engineering data as a set, let A be the set of all data processed during an engineering design process with n disciplines and $A_1 \dots A_n$ are the sets of data processed by each particular discipline. Expressing this situation by the means of the set theory results in the equation

$$\begin{aligned} A &= A_1 \cup A_2 \cup \dots \cup A_n \\ &= \bigcup_{i \leq n} A_i \end{aligned}$$

Figure 4.5 illustrates the data sets of four engineering disciplines depicted as four coloured squares A_1, A_2, A_3 and A_4.

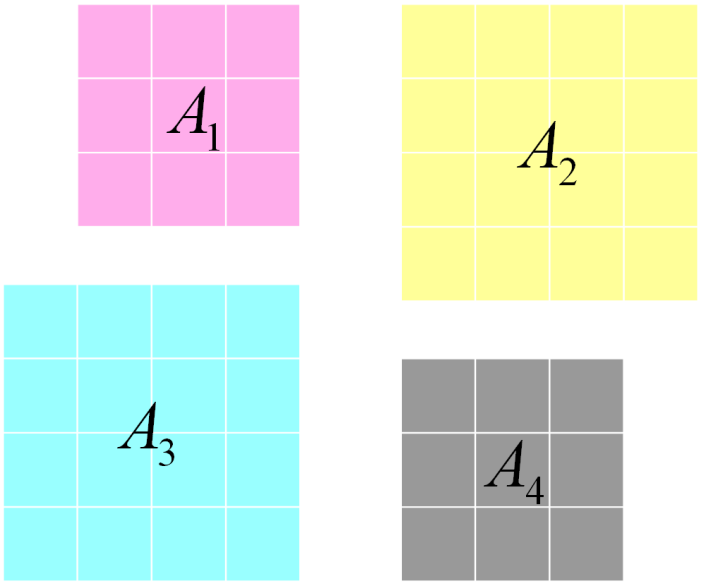


Figure 4.5.: An example of 4 data sets, A_1, A_2, A_3 and A_4, illustrated as 4 squares, representing the data processed by 4 different engineering disciplines

The amount of data processed is different for each discipline or in other words, the data sets $A_1 \dots A_n$ are of different cardinality. The squares A_1 and A_4 in Figure 4.5 are of the same size or, in other words, have the same cardinality as well as A_2 and A_3. So, in this example the disciplines A_2 and A_3 process a larger set of data than A_1 and A_4.

The data sets $A_1 \ldots A_n$ of the different engineering disciplines overlap, since they are based on each other's work and exchange data during the design process. Therefore, the cardinality of A, the set of all data processed during the whole design process, is smaller than the sum of the cardinalities of the data sets processed by the individual engineering disciplines.

$$|A| < |A_1| + |A_2| + \ldots + |A_n|$$
$$\Rightarrow |A| < \sum_{i=1}^{n} |A_i|$$

For the above example, Figure 4.6 a) and b) illustrate how the data setsA_1, A_2, A_3 and A_4 overlap each other and form the total data set A.

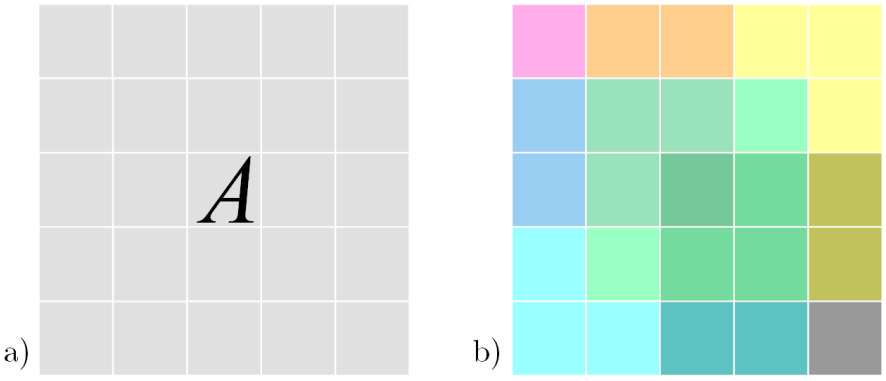


Figure 4.6.: (a) A represents the set of all data processed during the engineering process (b) Overlapping data sets A_1, A_2, A_3 and A_4 within A

There is one data set common in all data sets of the engineering disciplines involved in the design process, which overlap with all four other subsets. This data set is referred to as C the common or *core data set*.

$$C = A_1 \cap A_2 \cap \ldots \cap A_n$$
$$= \bigcap_{i \leq n} A_i$$

Hence, the data set processed by one engineering discipline can be divided into three subsets: the common/core data set C, the discipline specific data D and the data set E containing the data exchanged with other engineering disciplines.

$$A_i = C \cup D_i \cup E_i$$

The data set E consists of m *detailed design data modules* $M_1 \ldots M_m$.

$$\begin{aligned} E_i &= M_1 \cup M_2 \cup \ldots \cup M_m \\ &= \bigcup_{j \leq m} M_j \end{aligned}$$

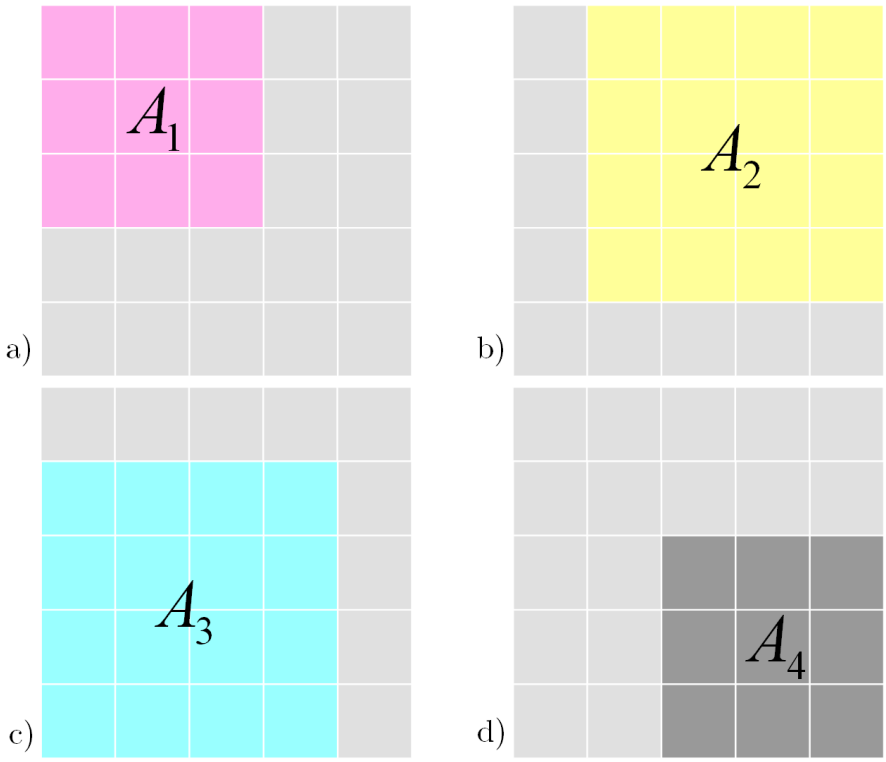


Figure 4.7.: A_1, A_2, A_3 and A_4 as subsets of A

Figure 4.7 illustrates how the data sets A_1, A_2, A_3 and A_4 form the subsets of A the total set of data processed during the engineering design process.

How the datasets A_1, A_2, A_3 and A_4 overlap in the total data set A is illustrated in Figure 4.8 with the example of A_3. The overlapping data is contained in the detailed design data modules.

Basically, there are intersections which A_3 only has with one other discipline, either A_1, A_2 or A_4.

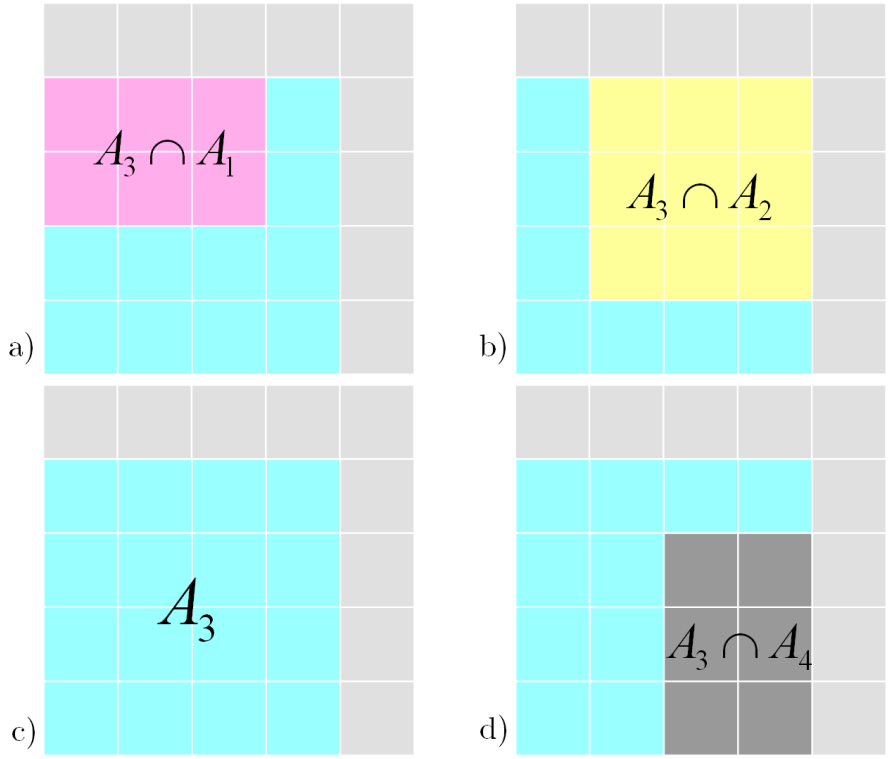


Figure 4.8.: Data Set Intersections of A_3

$$M_1 = A_1 \cap \bar{A}_2 \cap A_3 \cap \bar{A}_4$$
$$M_2 = \bar{A}_1 \cap A_2 \cap A_3 \cap \bar{A}_4$$
$$M_3 = \bar{A}_1 \cap \bar{A}_2 \cap A_3 \cap A_4$$

However, the intersections of A_3 with the other data sets again overlap as illustrated in Figure 4.9. There are intersections which A_3 commonly has with two disciplines.

$$M_4 = A_1 \cap A_2 \cap A_3 \cap \bar{A}_4$$
$$M_5 = \bar{A}_1 \cap A_2 \cap A_3 \cap A_4$$

Each of these modules contains the overlapping data of the discipline i with another discipline or a group of other disciplines. The maximal number of modules can be calculated with the binomial coefficient where n is the total number of disciplines involved in the engineering design process and k is the number of disciplines sharing a specific module.

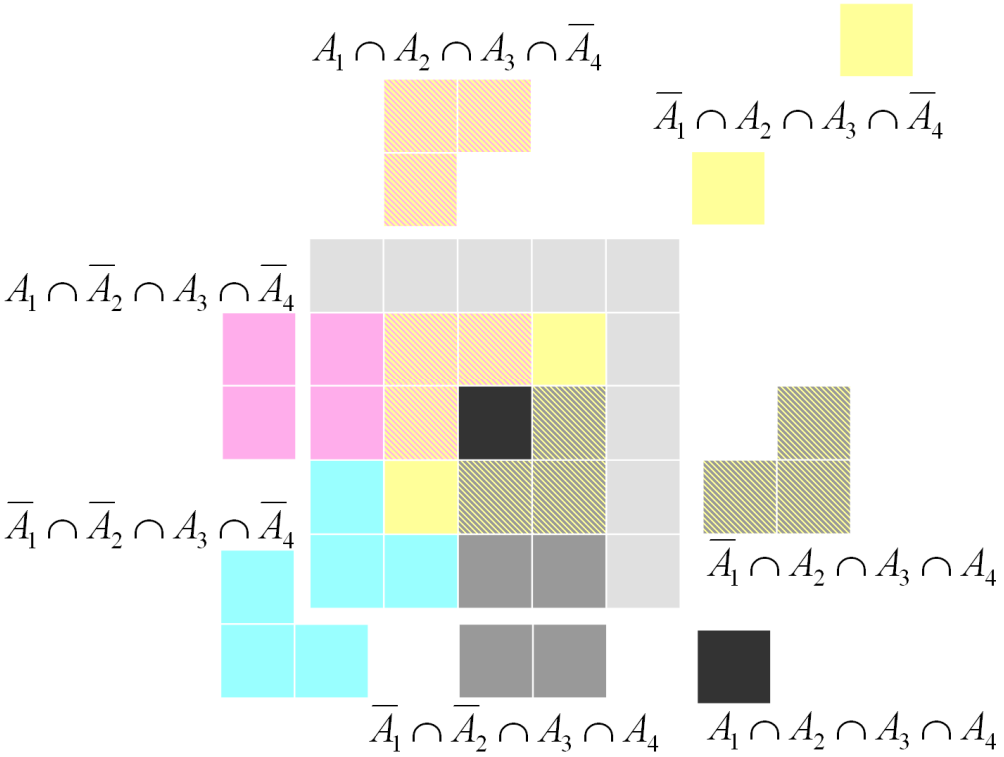


Figure 4.9.: Data Modules

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

In the case of the example given in this section $n = 4$ and thus, six modules exchanged between two disciplines ($k = 2$) and four modules exchanged between three disciplines ($k = 3$). In this way it is possible to determine the number and the content of the detailed design modules exchanged.

4.5. Summary

The core design data is identified by constructing the intersection of the data structures underlying the applied software tools in an engineering design process. This identification can be done by matching the nomenclature and hierarchical relations of the elements contained in the data structures, as demonstrated with reference to the mechatronic design process of body shop production lines. In this example the core data structure matches the basic functional structure of a body shop production line, meaning it is divided into PLC areas, safety circuits, operations, jigs and robots. Incorporating this core data structure in all applied software tools allows for seamless information exchange concerning this basic

functional structure which is relevant to all engineering disciplines involved.

Information which goes beyond the core design data is stored in detailed design data modules. Such information is typically not relevant to all engineering disciplines, only two or three. Information stored in a detailed design data module can be adapted to changes in the process because only a few disciplines need to agree on format and content. Additionally, the format and content of a detailed design module can be adapted to the technical possibilities and constraints of the import and export interfaces of the software tools producing and consuming this information.

Besides the specific application of the middle-in data modelling approach to the engineering design process of body shop production lines, set theory allows for a general mathematical approach to determine core design data and detailed design data modules. This is a convenient way to determine general characteristics of the data exchange situation arising during interdisciplinary engineering design processes.

5. Implementing the Middle-in Data Modelling Approach

The previous chapter introduced a concept for software systems integration in a mechatronic engineering design process based on a new middle-in data modelling strategy. This chapter demonstrates how to implement a systems integration solution based on this new strategy in an industrial mechatronic engineering design process.

Because this new concept is focussed on data modelling, it can be realised using various implementation technologies, such as standard file formats or central data bases. The decision about which implementation technology to choose very much depends on the surrounding conditions, especially the available financial resources and the IT infrastructure.
The implementation realised in this research project is intended to be applicable to a variety of mechatronic engineering design processes. Thus, it is comprised of very elementary implementation technologies with low requirements on financial resources and IT infrastructure. Additionally, simple implementation technologies allow the users themselves to implement, maintain and further develop the integration technology. Accordingly, they can adapt the integration solution to their requirements. The knowledge and insights gained should also be applicable to more technically complex integration solutions.

The architecture of a systems integration solution based on a middle-in data modelling strategy is illustrated in Figure 5.1. The primary task of such a systems integration solution is to exchange both core and detailed design data. This task can be divided into four major groups. The first is the group which imports core design data from the software tools in use; it also opens *Core Design Data Projects* on the basis of previously imported and provided data. In the second group detailed design data modules are added to the core design data on the basis of *Templates of Detailed Design Data Modules*. These templates are defined by the third group.

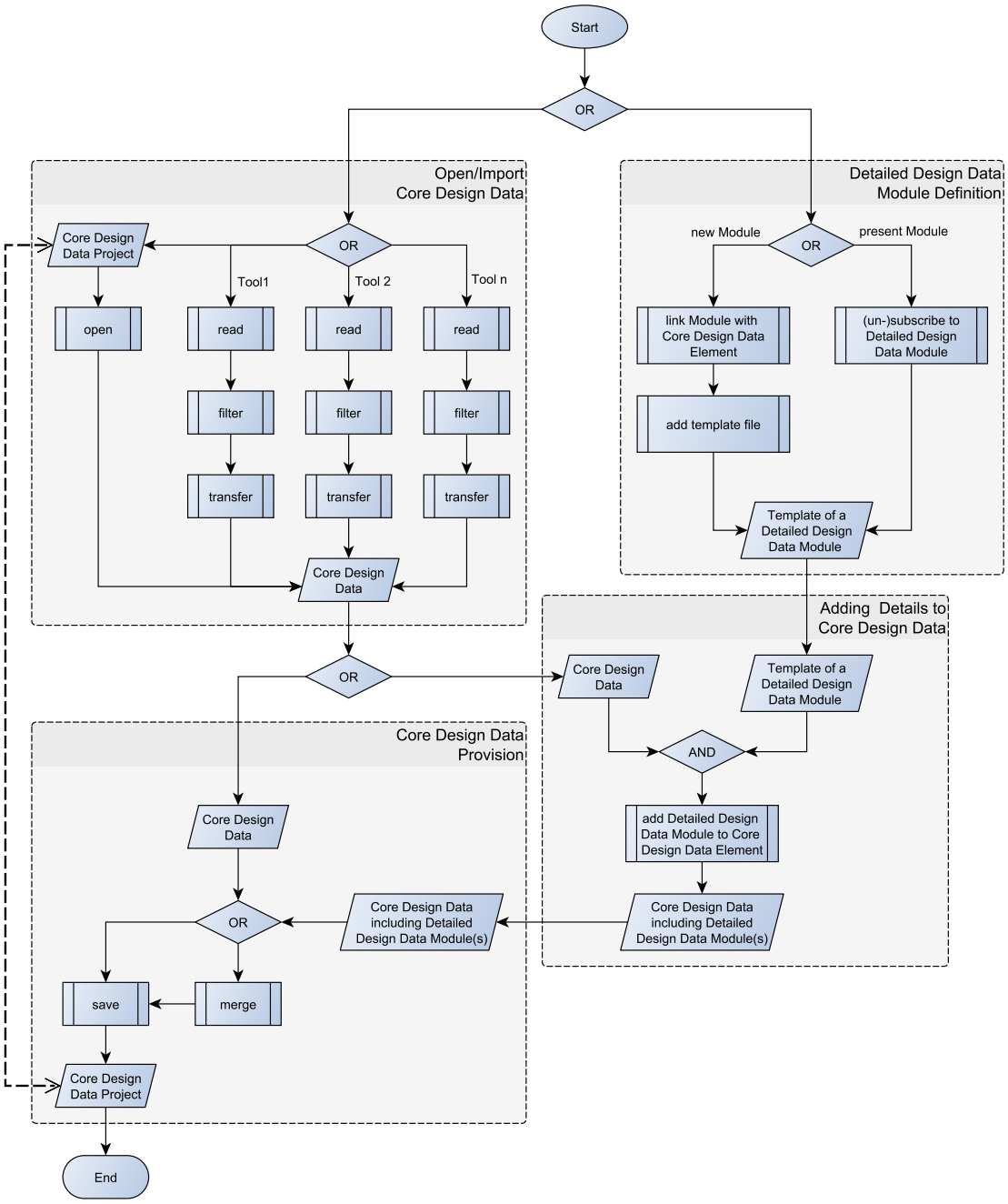


Figure 5.1.: Architecture of a systems integration solution based on middle-in data modelling

The fourth and last group provides the core design data or the enriched core design data including detailed design data modules to the other engineering disciplines involved in the interdisciplinary engineering design process. This chapter contains four sections according to the mentioned four major groups of the systems integration solution architecture.

5.1. Importing and Opening Core Design Data

A *Core Design Data Project* can directly opened. When no core design data project is present then the core design data can be imported from one of the other software tools in operation.
The import sequence for data from every software tool in use is set up similarly and consists of three procedures:

- to **read** data from a particular software tool, reducing it to the relevant information;
- to **filter** elements of the core design data from the relevant information;
- and to **transfer** relevant elements into a common format.

The *read* sub-procedure accesses the data underlying the software tool involved. This sub-procedure may directly access a database, a file or it accesses data with the aid of a specific API for remote procedure calls. Moreover, the *read* sub-procedure focuses on the parts of the underlying data which contain the core design data and ignores irrelevant data. It identifies the name of the element, the original identifier and the original data type.
Once the relevant content has been read, the elements of the manufacturing system are filtered according to the core design of the line in the *filter* sub-procedure which identifies the elements of the core design data from the read data. To carry out this procedure, it contains a query for each element of the core design data. These queries include string compare instructions for the attributes of the read elements.
The *transfer* sub-procedure transfers the read elements with their attributes to core design data elements according to certain transfer rules. Incidentally, it is necessary to align different spellings in the different disciplines. There is one transfer rule defined for each core design data element.

The *read* procedure is unique for each software tool used. The *filter* and the *transfer* procedures consist of one filter rule respectively for each core design data element.

When the core design data is extended by an additional element, the filter and transfer procedures need to be extended by an additional filter and transfer rule. The previous *filter* and *transfer* rules remain untouched, ensuring the work required to introduce additional elements to the core design data is kept to a minimum.
The modular architecture of the import procedure makes it possible to reduce the required effort to maintain and extend it. When a new software tool is introduced to the process or modifications of software tools in use are made affecting the data format, a new *read*, *filter* and *transfer* procedure needs to be implemented without affecting the implementation of the import sequences for data from the other software tools.

5.2. Detailed Design Data Module Definition

A detailed design data module contains data which goes beyond the core design data. It is distributed at a certain point in the engineering design process among a few engineering disciplines. The engineering discipline producing the information is responsible for defining the new detailed design data module. This definition includes a link to the module with the core design data element which is further described in detail by the information stored in the module.

Besides linking the detailed design data module with the core design data, the engineer responsible for it must also add a template file. This template file is an empty file which does not yet contain any detailed design data but is prepared in order to gather detailed design data and it describes the structure of a detailed design data module. For example, if the detailed design data module is a spreadsheet then the template file already contains the table with all headings and a definition of which data types are to be kept in the respective columns. The template does not contain any detailed data, however, as the lines of the table are still empty.

Depending on the template, a detailed design data module can either be produced by manually filling in the information or it can be produced automatically. The latter can be done by an export script from the particular software tool storing the information since the template formally describes the information stored in the detailed design data module. It is possible to implement transfer algorithms to convert the information into the form defined by the template and to extract the information for further processing. Regardless of whether manual or automatic processing is used, the template file supports engineers to monitor

the completeness of their work: as long as there are still blanks in the template file then the work has not yet been completed and further details have to be designed.

Engineers can subscribe to a detailed design data module when their work builds up on the information stored inside. Knowing the subscribers of the information stored in a module makes it possible to automatically inform them when changes occur connected to the information stored within. It also makes the data exchange process more transparent because it is clear which engineering discipline builds up on which information from which other discipline.

5.3. Adding Details to Core Design Data

When adding detailed design data modules to the core design data, it is required that the procedure as described above has been accomplished, meaning that core design data has been imported from a software tool being used or a present core design data project has been opened. Furthermore, at least one template of a detailed design data module must have been defined.

Adding a detailed design data module to a core design element establishes a link between both sets of information. Technically, a variety of options exist to establish this link. Which alternative is chosen depends on the implementation method. For the implementation undertaken in this research project, the file containing the information on the detailed design data module is stored in the respective folder representing the core data element.
This file can be manually copied to the respective folder but automatic processing is also possible. The hierarchical setup of a production line which is contained in every data model underlying the software tools is represented in the folder structure. The file name is set according to its template file, so, the software involved can automatically copy the information to the appropriate folder where it can be found by other software tools for further automated processing.

5.4. Core Design Data Provision

A core design data project can be provided to the disciplines involved in the engineering design process both with and without detailed design data modules. If a present core design data project has already been saved at the position where the new project is supposed to be saved, then the two projects are merged. For that

purpose a two-way merge algorithm is applied comparing the present and the new core design data projects. Elements which have been appended or deleted in the new project are presented to the user for revision and to correct the changes where necessary.
Once the changes are accepted, an automated notification mechanism is initiated. Every user who has subscribed to the particular template of the module is notified that there has been a module of their interest added or removed to or from the core design data project.

6. Case Study

This chapter presents the case study implementing the middle-in data modelling strategy to the design of body shop production lines for the automotive industry as a typical example of an industrial mechatronic engineering process. Therefore, the first section presents the experimental setup of the case study including the surrounding conditions of body shop production line design. In the second section, the software prototype is presented which was introduced to the engineering design process of body shop production lines to implement the middle-in data modelling strategy for a systems integration solution.

6.1. Experimental Setup

This section presents the characteristic body shop production lines and provides an introduction to their mechatronic engineering design process. This includes an illustration of the engineering records and documents produced throughout the design process. Moreover, the software tools applied to create these records and documents are presented with respect to their degree of integration and the way they are able to exchange data.

The new concept for systems integration based on a middle-in data modelling strategy has been implemented in the tools and plant equipment building division of the AUDI AG and the mechatronic engineering design process of its body shop production lines. These production lines are a typical example of highly automated, custom-build and special purpose machines since they are planned, designed and commissioned only once, for a specific car series in a specific plant. A body shop production line project has a long lead time, between two and three years, from the first layout until commissioning is complete and the line has begun series production (Kiefer et al., 2006).
For the design of body shop production lines, the mechatronic disciplines mechanical and electrical engineering as well as software development need to work together closely. To accomplish this, the software tools employed by the different disciplines

must be able to exchange data seamlessly. Some software tools are well integrated, while others can hardly exchange any data.

To enable users to introduce, maintain and enhance the integration solution, the developed software tool builds upon well-known technologies such as spreadsheet tables and folder hierarchies in a file system as well as simple emails. This enhances usability for engineers by limiting the required specialist software integration knowledge, particularly the integrated development environment, the data storage and the user management as much as possible.
The software tool developed in this research project, created with the aid of Mircrosoft Visual Studio 2005, is called the "ProcessExplorer". The advantage of this software tool is that applications with Windows look and feel can be produced with this environment, which should increase user acceptance of the software tool.

The ProcessExplorer stores all data in a shared folder on a network device where it can easily accessed by all involved engineers. A shared folder on a network device has the advantage that, in contrast to a database, it can be more easily surveyed by users, who understand its technical limitations because they use this technology daily.
The ProcessExplorer uses the built-in Windows user management, so it is not required to have a distinct user name and password. The user management works automatically in the background.

6.1.1. Characteristics of Body Shop Production Lines

Figure 6.1 shows the layout of a typical body shop. It serves as an example to explain the general structure of a body shop which varies in required space depending only on the target number of produced units.
Contrasting colours are used to distinguish between the four major sub-assemblies of a body shop: underbody sub-assembly, body shell sub-assembly, body side sub-assembly and hang-on-parts sub-assembly. These sub-assemblies are arranged so that the assembled parts can be seamlessly transferred to the successor sub-assembly.

Each sub-assembly is divided into minor sub-assemblies, so-called stations. The underbody sub-assembly contains stations producing front floor and the rear floor as well as the side member. These body parts are transferred to the underbody assembly line where they are joined together. Then the underbody is transported to the body shell assembly line. Here the side panels are joined with the underbody

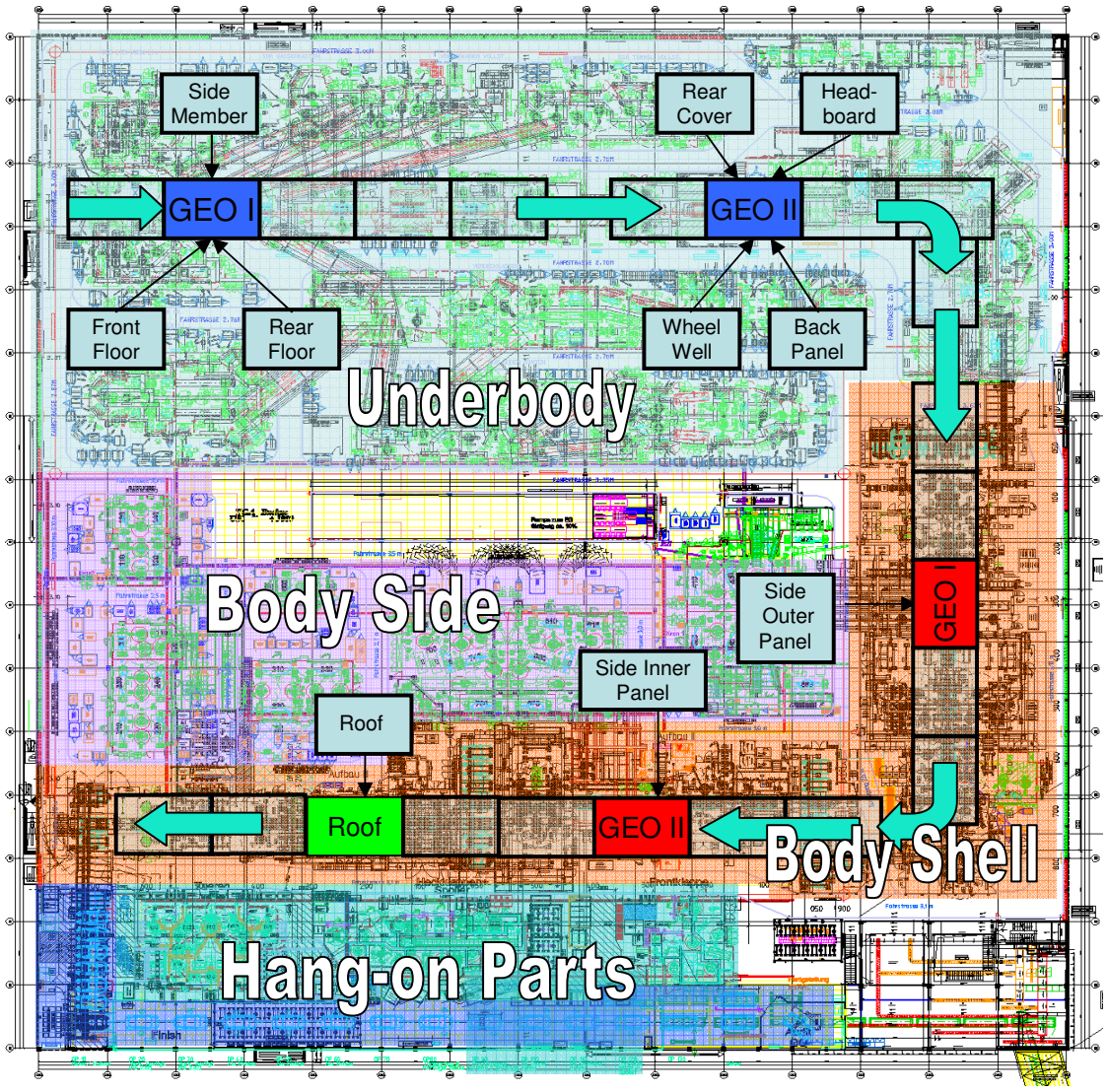


Figure 6.1.: Typical body shop layout

in so-called geometry stations (GEO stations). The hang-on-parts sub-assembly contains stations producing the tailgate, the bonnet and the doors. These parts are also conveyed to the body shell sub-assembly where they are attached to the body in white.

6.1.2. Engineering Records for the Design of Body Shop Production Lines

Pneumatic schematics, electrical schematics as well as PLC and robot programs are the deliverables which are passed from one process step to the next. These results are summarised under the generic term of *engineering records and documents*. To analyse the cooperation of the engineering disciplines, it is important to understand the content and extent of these engineering records and documents. They are described in this subsection with the aid of a few examples.

6.1.2.1. Pneumatic Schematics

The process step of pneumatic design produces schematics which are passed on to the process step of electrical design. Moreover, pneumatic schematics are the basis for the pneumatic installation.
There is always one set of pneumatic schematics for each operation within a body shop station with penumatic actors, mostly for a jig or a gripper. The pneumatic schematics document how pneumatic components are provided with compressed air. They contain an overview page showing a 2D view of the gripper or jig and labels of all fixtures, valves and sensors. In addition, an assignment list is contained which associates sensors and actors with its norm-description and its manufacturer. Moreover, the pneumatic schematics contain a sequence diagram summarising the sequence of an operation.
The structure of the schematics is standardised as well as the set of pneumatic components to realise pneumatic functionality. In case of failure, the maintenance personal do not need to think their way into each body shop station anew. This leads to shorter downtime of body shop stations. Besides this, the usage of a compulsory set of pneumatic components allows the purchasing department to order them in quantity and gain better prices and the storage of replacement parts is simplified. The extract of pneumatic schematics on Figure 6.2 shows three fixture groups controlled by electromagnetic 5-ports/2-way valves and throttle check valves.

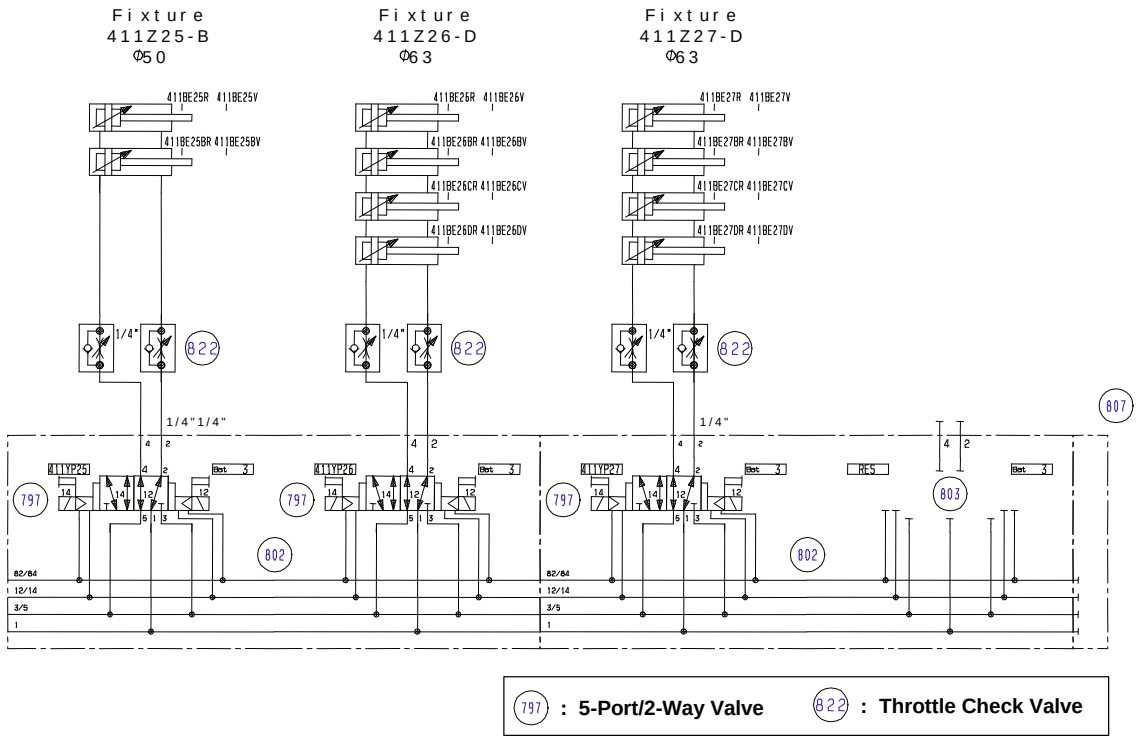


Figure 6.2.: Extract of pneumatic schematics

6.1.2.2. Electrical Schematics

The electrical schematics are exchanged between electrical design and PLC software design. It builds the basis of the electrical installation of a body shop production line and is an essential part of a production line's documentation for fault tracing and extension of its the functionality.

One set of electrical schematics comprises one PLC area within a body shop station. It illustrates how the electrical components are connected with each other. It contains several hundred pages, depending on the complexity of the station.

The structure of the electrical schematics and the components applied are also standardised like the pneumatic schematics. This includes, for example, a standardised structure of a switch cabinet, a compulsory part list as well as a standardised structure of the electrical schematics. This has the same advantages concerning maintenance, purchasing and storage of replacement parts as in pneumatic design.

Figure 6.3 is an extract of electrical schematics. It shows an electrical engine used to accurately position the body in the body shop station.

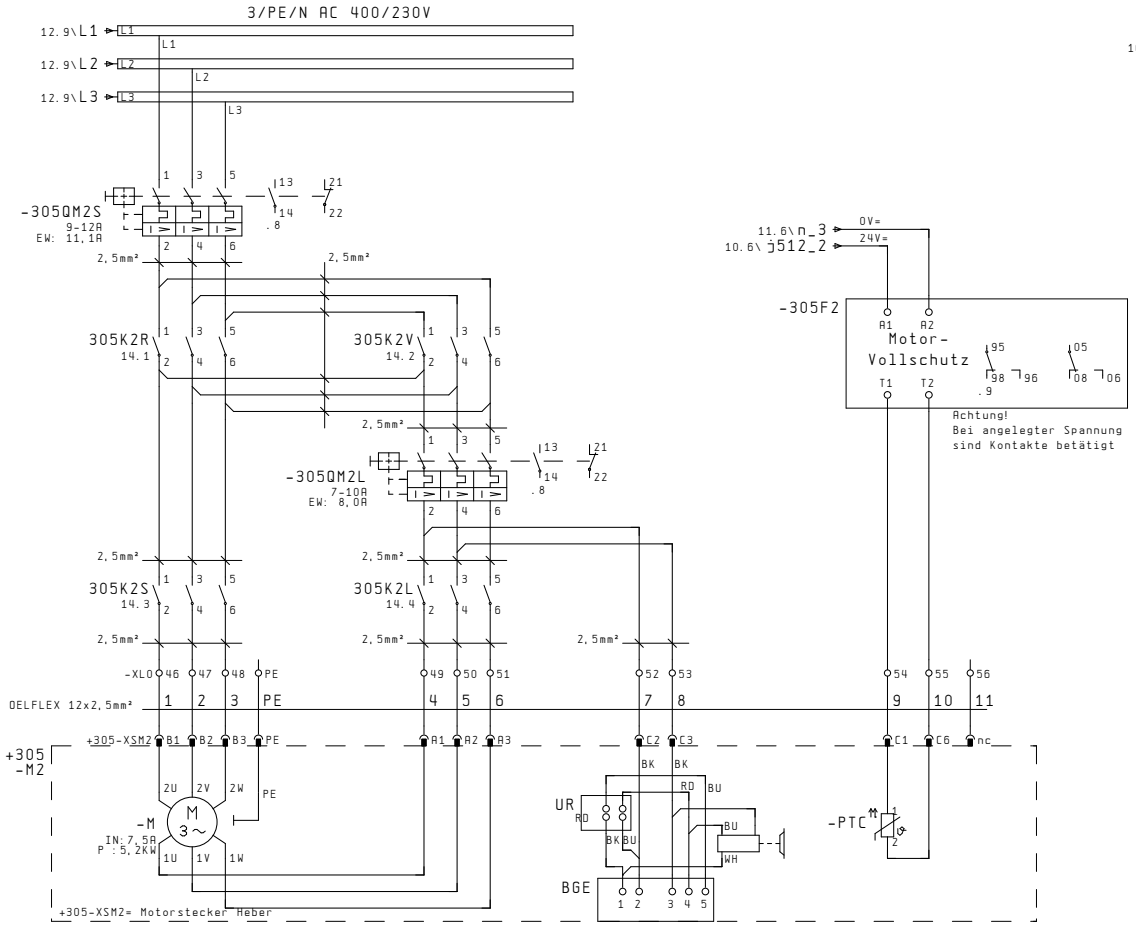


Figure 6.3.: Extract of electrical schematics

6.1.2.3. PLC Software

The PLC software is not only required for the documentation of the production line but it is essential for the functionality of the production line. This is the major difference between PLC software and pneumatic or electrical schematics.
In IEC 61131-3 three programming languages are defined for PLC software creation: Instruction List (IL), Ladder Diagram (LD) and Function Blocks (FB) (*IEC 61131-3*, 2003). IL is a pure text-based programming language while LD and FB are graphical programming languages. FB displays logic blocks with defined in- and outputs. LD was used as the standard programming language for PLC programs in our case study.

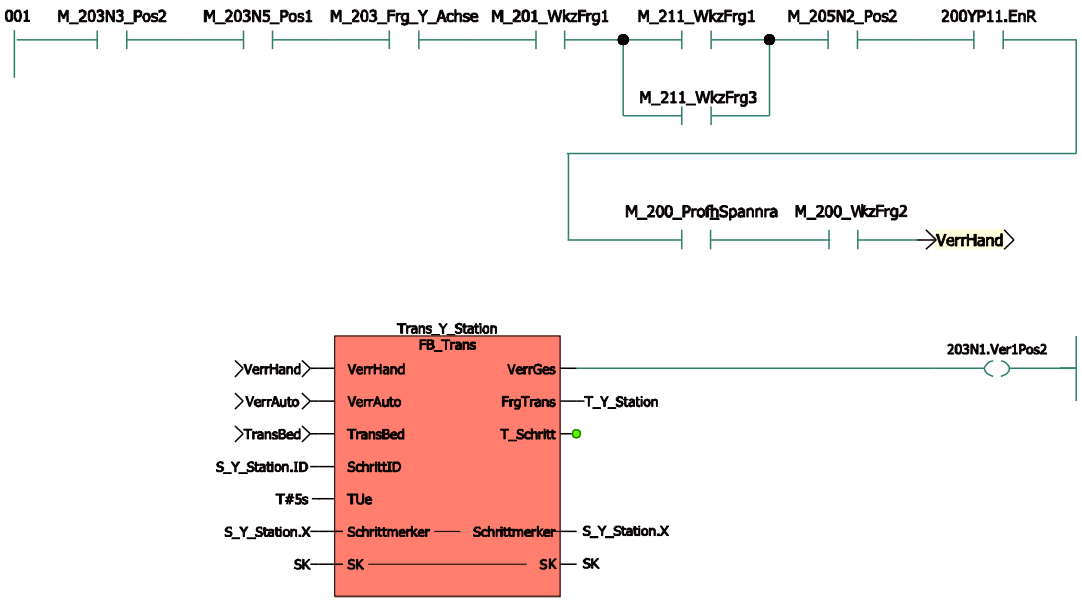


Figure 6.4.: Extract of a PLC program: parameterization of a function block with an operation defined as ladder diagram

Figure 6.4 shows an extract of the PLC program of a body shop station. It contains a function block in which the input `VerrHand` is parameterized by a logical operation defined as a ladder diagram. This operation is read from left to right, a gap in the line represents a signal. Several signals in a row are connected by a logic AND, parallel branches represent signals connected by logic OR. A slash placed inside a gap symbolises a negative query of the signal. The last signal on the line represents an output or a variable saving the result of the logic operation.

The sequence of a body shop station is programmed and controlled with the aid of a sequencer. The transfer from one sequence step to the next is undertaken depending on a transition condition. An example for a step sequence is given in Figure 6.5.

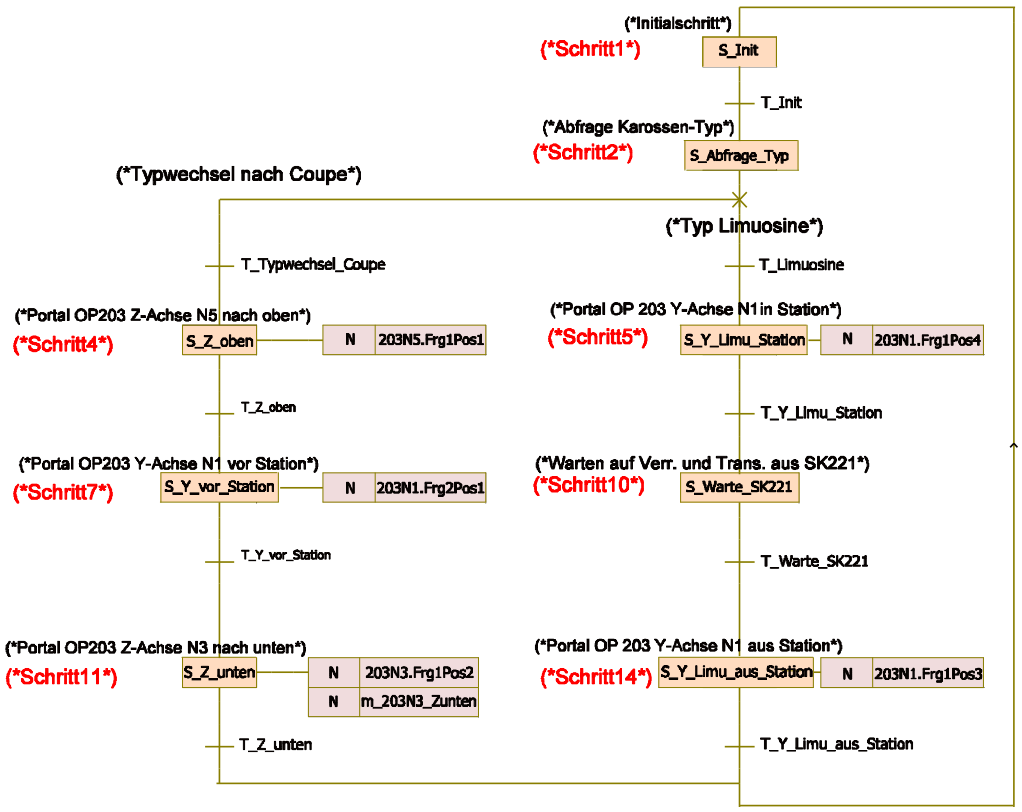


Figure 6.5.: Sequence in a PLC program

6.1.2.4. Robot Software

The robot software is one of the final products of the engineering design process like the PLC programme. It is also essential for the function of the production line.
A robot program is structured into one main program and several sub-programs that are called by the main program. A program contains a sequence of points which build the path of the robot. Reoccurring paths such as the path to the welding gun cap sharpener are stored in a sub-program and called by the main program. Besides the path, the robot program contains other logic. Signals can be set, queried or waited for. These signals control the instruction exchange with the PLC, with other robots and with the tool attached to the robot. Signals sent by the PLC to the robots include instructions to move to a specific point or to perform a certain action. Robots exchange interlocking signals among each other, when they work in overlapping areas. The signal exchange between a robot and its attached tool contains instructions to perform a certain action for example *close welding gun*. An example of a robot main program is given in Figure 6.6.

```
P1      PTP VB=10% VE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] P
        A849 = EIN
        SPSMAKRO0 = EIN
        -- Folge1 OP1120R01 --
P2      PTP VB=10% VE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] PU
        FB PSPS = EIN
        A852 = AUS
        SPSMAKRO120 = EIN
        t1 ( EIN ) = 0[1/10Sek]
        -- Roboterverriegelungen freigeben --
        A857 = EIN
        A858 = EIN
        A861 = EIN
        A862 = EIN
        -- Anlagenverriegelungen freigeben --
        A874 = EIN
        A876 = EIN
        A878 = EIN
        A880 = EIN
        A849 = AUS
        WARTE BIS E852
        A852 = EIN
        F66 = AUS
        FB PSPS = E845 & E853 & M49
        -- Aufruf UP101 Kappenfraesen --
        UP101 = E194 + E196 + E907
        -- Aufruf UP1 Schweissen Verst.spiegel Tvli 210 OP1121 --
        -- Aufruf UP2 Entnahme Verst.spiegel Tvli210 OP1121 --
        -- Aufruf UP3 Schweissen Tvli 210 OP1131 --
P3      PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s] PU
        FB PSPS = EIN
        A852 = AUS
        WARTE BIS E880
        A880 = AUS
        WARTE BIS E852
        A852 = EIN
        FB PSPS = E845 & E853 & E880 & M49
        UP1 = EIN
        UP2 = EIN
        UP3 = EIN
        -- Gesamtfertigmeldung setzen --
        -- Aufruf UP101 Kappenfraesen --
P4      PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTrig=5[1/100s] PU
        FB PSPS = EIN
        A852 = AUS
        WARTE BIS E200
        SPSMAKRO15 = E200
        WARTE BIS E852
        A852 = EIN
        F66 = AUS
        FB PSPS = E845 & E853 & M49
        -- Aufruf UP101 Kappenfraesen --
        UP101 = E194 + E196 + E907
P5      PTP VB=100% VE=0% ACC=100% Wzg=1 SPSTrig=0[1/100s] P
        FB PSPS = EIN
        FB PSPS = E845 & E853 & M49
        t1 ( EIN) = STOP
        t2 ( EIN ) = t1[1/10Sek]
        t2 ( EIN) = STOP
```

Figure 6.6.: Example of a robot main program

6.1.3. Engineering Disciplines and Software Systems Involved

The different software systems used by the engineering disciplines which participate in the engineering design process of body shop production lines are shown in Figure 6.7.

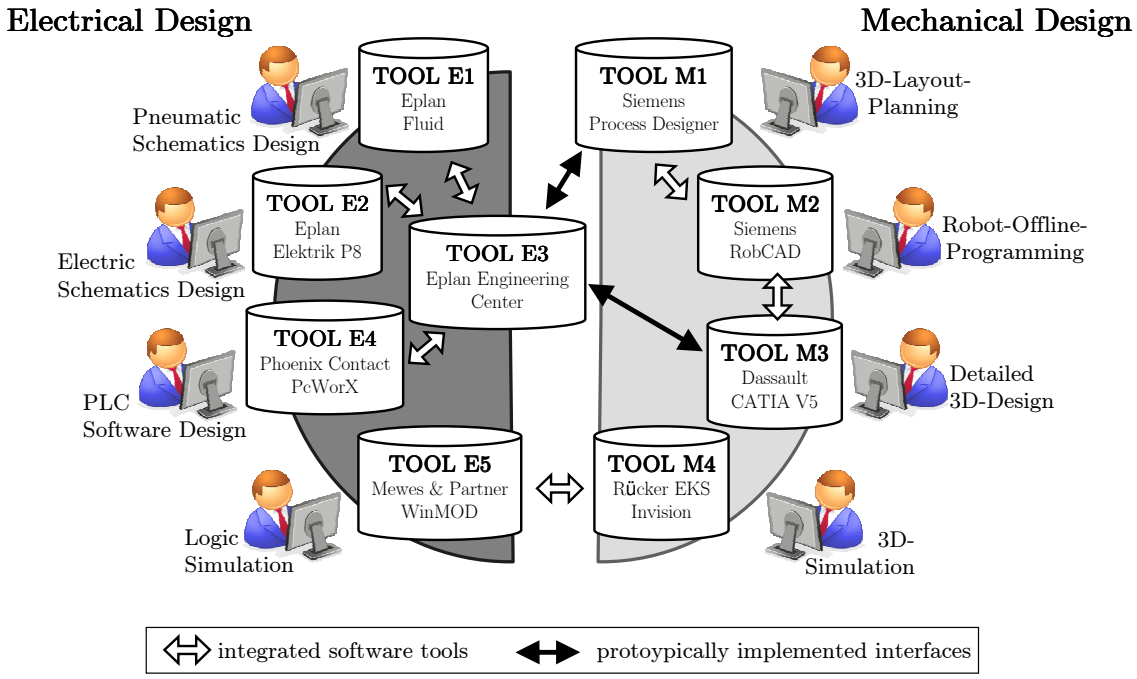


Figure 6.7.: Software tools used in mechatronic engineering design of body shop production lines

The division of the engineering disciplines into electrical and mechanical design also applies to the software landscape. Consequently, the software landscape is also characterised by two integrated environments: (1) the Siemens Tecnomatix tool suite used with *ProcessDesigner*, *RobCAD* and *ProcessSimulate* in mechanical engineering; (2) EEC with *Eplan P8 Electric/Fluid* and *PCWorX* for electrical engineering. The software tools within each of these two environments are well integrated whereas data exchange between these two integrated environments is nearly impossible.

Next, the design task undertaken during the phase of mechanical and electrical design as well as during virtual commissioning are introduced. The nature of body shop production line design is emphasised as a mechatronic engineering process with a mature software landscape, where each discipline applies the software tool which best suites its needs and requirements.

6.1.3.1. Mechanical Design

During 3D layout planning the general structure of a body shop production line is set, primarily with a focus on the line's mechanical design. Electrical and software design are only considered as cost elements for the production line cost calculation. Several alternative 3D layouts of a body shop production line are developed and evaluated with respect to their estimated cost, cycle time and operational availability using Siemens UGS' *ProcessDesigner* (Siemens PLM Software, 2013, *b*).
During the phase of detailed 3D design, 3D CAD models of body shop elements, such as jigs, are designed in detail. To support this task, Delmia's 3D CAD Software *Catia V5* (Dassault Systèmes, 2013) is utilised in because it is also used for body shell design in the car development division and 3D models of the body shell can easily be exchanged between body shell design and jig design. Thus, the jig can be optimised using the original geometrical data from body shell design. In addition to detailed jig design, Catia is also used to test the producibility of a body part by analysing whether each joining spot (e.g. a welding spot) can be accessed by the joining tool (e.g. a welding gun).
The resulting 3D models form the basis of robot offline programming. In this sub-process the paths of the robots from one working point to another are planned and simulated in order to avoid collisions among the robots and between robots and jigs. The robot offline programs are created with the aid of *RobCAD*, developed by the Israeli company Tecnomatix which is now a subsidiary of Siemens AG (Siemens PLM Software, 2013, *c*). It is able to import 3D CAD models from Catia.

6.1.3.2. Electrical Design

Electrical design starts with pneumatic schematics design which builds upon information from detailed 3D design with regard to the type of cylinders, valves and final position switches to be used. Additional information about cylinder grouping and their sequence of action is also required.
Pneumatic schematics design is followed by electrical schematics design. Electrical schematics contain information about the structure and wiring of switching cabinets as well as the address assignment of sensors and actors in the field bus. This information is used by the PLC programmer in combination with the sequence diagram of the pneumatic schematics for the creation of the control program in the sub-process of PLC software design.

The pneumatic and electrical schematics, as well as the PLC program are in part generated automatically by the *EEC* (EPLAN Software & Service, 2013, *b*), which

populates the connected target systems *EPLAN Fluid* for pneumatic schematics design (EPLAN Software & Service, 2013, *c*), *EPLAN P8 Electric* for electrical schematics design (EPLAN Software & Service, 2013, *a*) and *PcWorX* for PLC software design (PHOENIX CONTACT, 2013).
The EEC contains the structural model of a body shop production line built of re-usable components stored in a library. These re-usable components are connected with fragments of the PLC program as well as fragments of the electrical and the pneumatic schematics. Using an elaborate formula-mechanism, the EEC pieces these fragments together into engineering documents.

6.1.3.3. Virtual Commissioning

During virtual commissioning the interaction of robots and the PLC program is tested and further improved. *Invision*, developed by Rücker EKS, is used for the 3D simulation of the production line and the robot behaviour (RÜCKER EKS GmbH, 2013). It sets up the robot offline programs and the detailed 3D models from RobCAD. The logic simulation of a production line is undertaken with the aid of *WinMOD*, developed by Mewes and Partner (Mewes & Partner GmbH , 2013). The PLC program, which is tested during virtual commissioning, runs on the same type of PLC which will control the future production line. The industry PC which runs the HMI of the PLC is also of the same type as will be used in the future production line.

6.2. Systems Integration Based on Middle-in Data Modelling

The case study presents the sequence initiating the functions of the software prototype according to its architecture as presented above. The aim of this case study is to demonstrate that the new concept is applicable to systems integration in an industrial engineering process. Moreover, the case study functions as a basis for the evaluation of the efforts and the benefits of implementing the new systems integration concept.
The following subsection illustrates how different import sequences have been incorporated into the engineering design process of body shop production lines in the case study. Building on this, the second and third subsections introduce two use cases demonstrating how the exchange of core and detailed design data has been realised.

6.2.1. Implementation of Import Interfaces

This subsection describes the procedure for implementing the import interfaces for ProcessDesigner and EEC data. On this basis the labour-intensiveness of implementing further import interfaces is evaluated. Both software tools offer XML-based export files which are used for the import interface.
According to the architecture of a systems integration solution based on a middle-in data modelling strategy, the import procedure is divided into three steps: first *read* the data from the respective software tool; second *filter* the data for relevant information; and third *transfer* it to the core data model (see Chapter 5.1). The import interface for EEC and ProcessDesigner data was implemented accordingly.

6.2.1.1. ProcessDesigner Import Interface

To carry out 3D layout design the ProcessDesigner from Siemens Tecnomatix is applied which offers an XML-interface to export the body shop production line project. The extract of the XML-file containing the data of a PLC area is shown in the listing in Figure 6.8.

```
<?xml version="1.0"?>
<PrZone ExternalId="PP-90c84f8b-3083-4abf-9919-4e44b54c1542">
    <name>PLC 4</name>
        <!--...-->
  - <children>
        <item>PP-409a0105-06d4-48e6-a1ef-024e0b121f4f</item>
    </children>
        <!--...-->
</PrZone>
```

Figure 6.8.: Extract of the ProcessDesigner export XML showing a PLC area

The most important parameters are `PrZone`, which indicates the original data type of the element. Further data types such as `PmLayout`, `Pm3DRep` or `PmImage`, contain data for displaying the 3D machine layout in the ProcessDesigner. For extracting the core design data containing the major elements and their hierarchical relations, the relevant data types are `PrZone`, `PrStation`, `CompoundResourceRobot` and `CompoundResourceFixture`. Precisely which core design data element is contained in the node can be seen from the `name` attribute. Each dataset in the XML-file can be clearly identified by its `ExternalId`. This export format does not use XML-hierarchies to nest underlying elements but an attribute `children` keeping the `ExternalId` of its underlying elements. In this way, safety circuits, operations, robots and jigs in the XML-file as illustrated in Figure 6.9 can be found.

```
<?xml version="1.0"?>
<Objects>
        <!--...-->
  - <PrZone ExternalId="PP-409a0105-06d4-48e6-a1ef-024e0b121f4f">
        <name>Safety Circuit 1</name>
            <!--...-->
      - <children>
            <!--...-->
            <item>PP-f82a1be9-86bf-4557-9837-8becaff72306</item>
            <!--...-->
        </children>
            <!--...-->
    </PrZone>
        <!--...-->
  - <PrStation ExternalId="PP-82fcd78f-e6ab-4a06-8f44-1c8c93f7b56a">
        <name>Operation 1030</name>
            <!--...-->
      - <children>
            <item>PP-ea2e70d9-8a0d-4be7-bfad-8d83e5f43c20</item>
            <item>PP-57d665df-f3dd-4023-848d-8e1bda887fdc</item>
            <!--...-->
        </children>
            <!--...-->
    </PrStation>
        <!--...-->
  - <CompoundResourceFixture ExternalId="PP-ea2e70d9-8a0d-4be7-bfad-8d83e5f43c20">
        <name>Jig</name>
            <!--...-->
    </CompoundResourceFixture>
        <!--...-->
  - <CompoundResourceRobot ExternalId="PP-57d665df-f3dd-4023-848d-8e1bda887fdc">
        <name>Robot 1</name>
            <!--...-->
    </CompoundResourceRobot>
        <!--...-->
</Objects>
```

Figure 6.9.: Hierarchy of elements in a ProcessDesigner export file

The filter sub-procedure for ProcessDesigner filters the data, for example, for elements of the `PrZone` data type with a name corresponding to a safety circuit. The transfer sub procedure then transfers the element name according to the defined naming conventions. It also saves the subordinated element and the `ExternalId` to link the core design data with its source elements.

6.2.1.2. Eplan Engineering Center Import Interface

In contrast to the ProcessDesigner Export, the EEC export uses XML functionality to store hierarchies to indicate that an element has further underlying elements. The hierarchical relation of the elements in the XML file refers to the relation between the elements of a body shop production line, so they can be associated one to one. However, the various elements do not differ from each other according to their data type, as in the ProcessDesigner export file where there was a differentiation between, for example, `PrStation` and `CompondRessourceRobot`. All relevant elements in the EEC data are of the type `com.mind8.mechatronic.skill.eos.MechatronicComponent` so the elements need to be distinguished by their names and their hierarchical relations. Figure 6.10 shows an extract of an EEC export XML file.

A component called `PLC` has an underlying safety circuit representing a corresponding element of the body shop production line. Underlying the safety circuit are several operations with robots and jigs.
Each element has a unique identifier which is stored in the `<ck>` element, like for example `<ck v="-7428952197789490365" />`. This identifier clearly differentiates two identically named elements which hierarchically underlie the same father element.

For all elements of the data type `com.mind8.mechatronic.skill.eos.MechatronicComponent`, the filter sub-procedure distinguishes among the relevant elements for the core design data by their names. The transfer sub-procedure of the EEC data then adjusts the name according to the naming conventions for the core design data element. It also saves the value of the `<ck>` element as a unique identifier for linking the core design data element with its source element in the EEC data.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- [...] -->
<c n="rootUnit">
  - <eo isa="com.mind8.mechatronic.skill.eos.ProjectRootPackage">
        <ck v="7034736303634103156"/>
        <a n="name" v="Doors_Front_Left"/>
            <!-- [...] -->
      - <c n="mechatronicRoot">
          - <eo isa="com.mind8.mechatronic.skill.eos.ProjectMechatronicRoot">
                <ck v="5725314632122786780"/>
                <a n="name" v="Mechatronic"/>
                    <!-- [...] -->
              - <c n="mechatronicComponents">
                  - <eo isa="com.mind8.mechatronic.skill.eos.MechatronicComponent">
                        <ck v="-7428952197789490365"/>
                        <a n="name" v="Doors_Front_Left"/>
                      - <c n="mechatronicObjects">
                          - <eo isa="com.mind8.mechatronic.skill.eos.MechatronicComponent">
                                <ck v="2991166962730491445"/>
                                <a n="name" v="PLC"/>
                              - <c n="mechatronicObjects">
                                  - <eo isa="com.mind8.mechatronic.skill.eos.MechatronicComponent">
                                        <ck v="-6222000172374936111"/>
                                        <a n="name" v="Safety_Circuit"/>
                                      - <c n="mechatronicObjects">
                                            <!-- [...] -->
                                        - <eo isa="com.mind8.mechatronic.skill.eos.MechatronicComponent">
                                              <ck v="-837614863656864509"/>
                                              <a n="name" v="OP1030"/>
                                            - <c n="mechatronicObjects">
                                                  <!-- [...] -->
                                              - <eo
                                                isa="com.mind8.mechatronic.skill.eos.MechatronicComponent">
                                                    <ck v="9097522583638579866"/>
                                                    <a n="name" v="Robot"/>
                                                </eo>
                                                  <!-- [...] -->
                                              - <eo
                                                isa="com.mind8.mechatronic.skill.eos.MechatronicComponent">
                                                    <ck v="-8522159546142983507"/>
                                                    <a n="name" v="Jig"/>
                                                </eo>
                                                  <!-- [...] -->
                                            </c>
                                        </eo>
                                    </c>
                                </eo>
                            </c>
                        </eo>
                    </c>
                </eo>
            </c>
        </eo>
    </c>
</c>
```

Figure 6.10.: An extract of an EEC export XML file

6.2.1.3. Conclusions Regarding Present Import Interfaces

The data provided by both software tools allow the determination of the core design data of a body shop production line. The XML-format enables a straight forward serialising and parsing of the relevant data. The human readability of XML data makes it possible to identify and match information of a production line component even when different modelling techniques are applied to describe similar physical aspects, for example hierarchical relations of the production line components. The import algorithm for the respective data can be adapted accordingly.

6.2.2. Exchange of Core Design Data

In general, this case includes the initial generation of core design data using a software tool and the modification of core design data by a second software tool. The extraction of the core design data from the first software tool and merging it with the data from the second software tool is accomplished with a third software tool, the ProcessExplorer, which has been developed in this research.The use case is outlined in Figure 6.11. It involves the exchange of core design data between a 3D layout designer and the electrical designer.

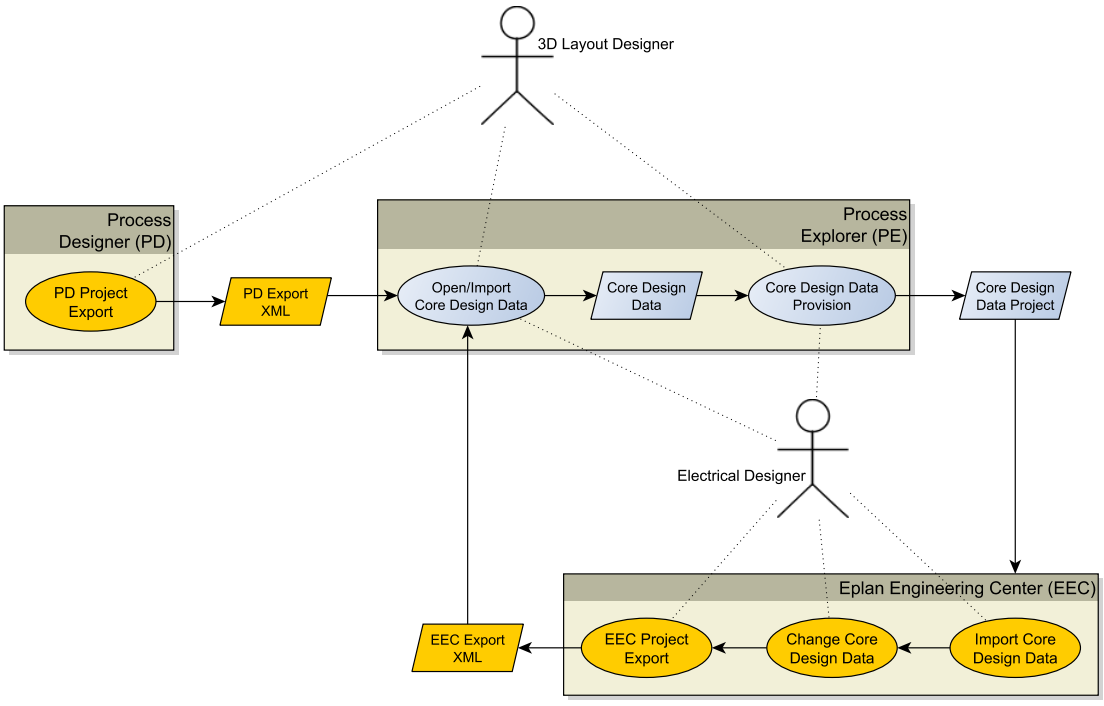


Figure 6.11.: Use case: exchange of Core Design Data

The 3D layout design includes the initial set up of a basic quantitative structure of a body shop production line and describes which component is contained in which number including their basic hierarchical relations.

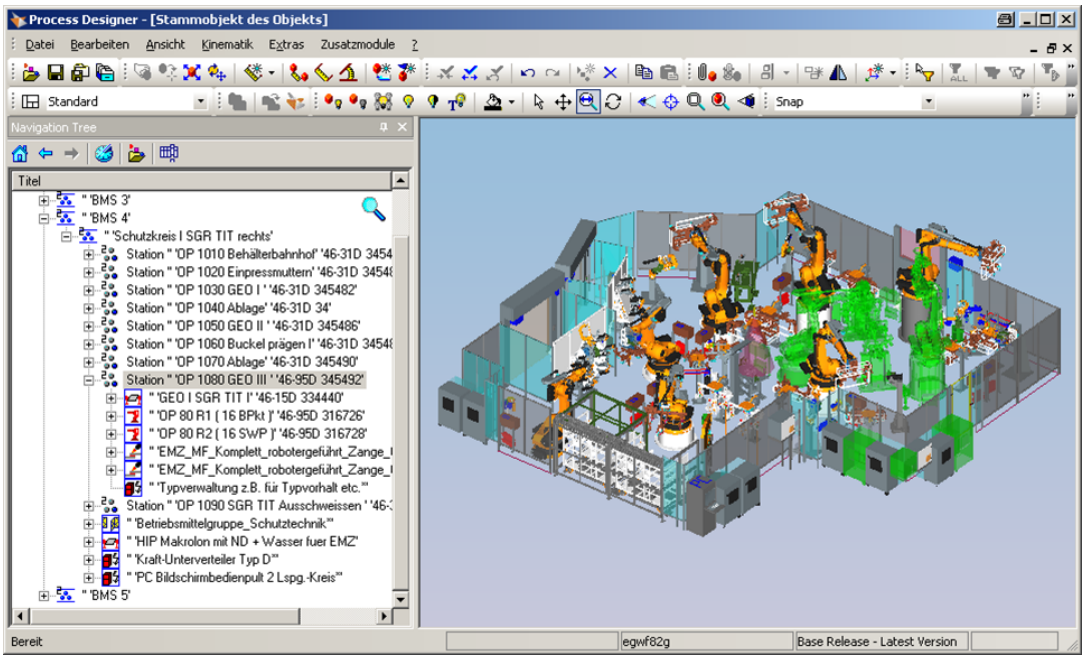


Figure 6.12.: Structure of a body shop production line in the ProcessDesigner

Figure 6.12 shows a screenshot from the ProcessDesigner during the 3D layout design of the body shop doors assembly line, a sample project for this case study. The ProcessDesigner project containing all 3D layout data for one specific production line can be exported from the ProcessDesigner in one XML file which can be imported by the ProcessExplorer. As described in the architecture chapter, the XML file is first read by the ProcessExplorer, then the relevant core design information is filtered and transformed to a standard format. A completed import by the ProcessExplorer is shown in Figure 6.13.

Then, the ProcessExplorer saves the data exported by the ProcessDesigner as a core design data project on a network device where it can be accessed by all the other disciplines involved in the engineering design process. A corresponding folder structure is then set up in a temporary folder on the local hard disk. This folder structure corresponds to the hierarchical relations of the major elements of a body shop production line and each folder represents one element and each sub-folder represents its underlying elements.

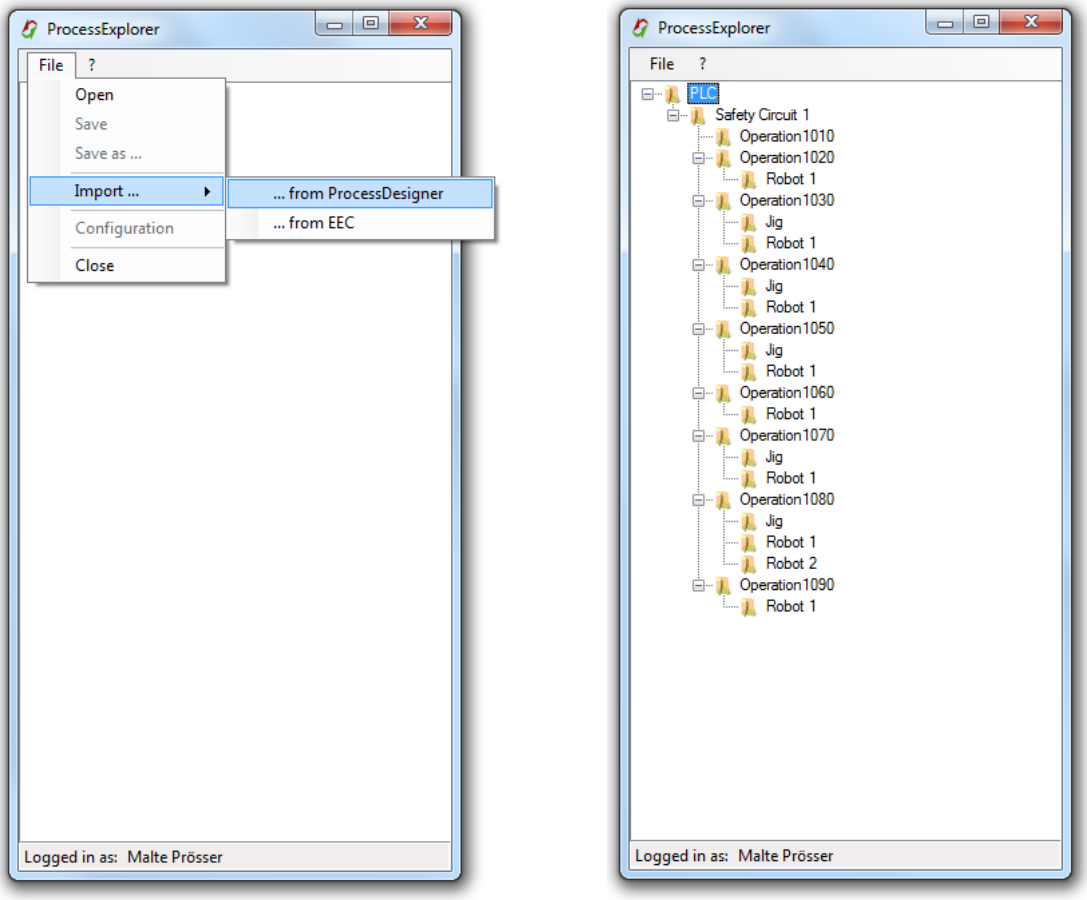


Figure 6.13.: Import core design data from ProcessDesigner

In addition to saving the core design data project as a folder structure, it can be stored in any form necessary for the software tools being used. For this case study ProcessExplorer also provides the data as a Comma-Separated Values (CSV) file because the EEC has an import interface for this file format. It is named and stored in the same position in the folder structure corresponding to the body shop production line project. Thus, it can be referred to by the EEC automatically and the amount of work necessary to create an import interface can be reduced. The imported data in the EEC is shown in Figure 6.14.

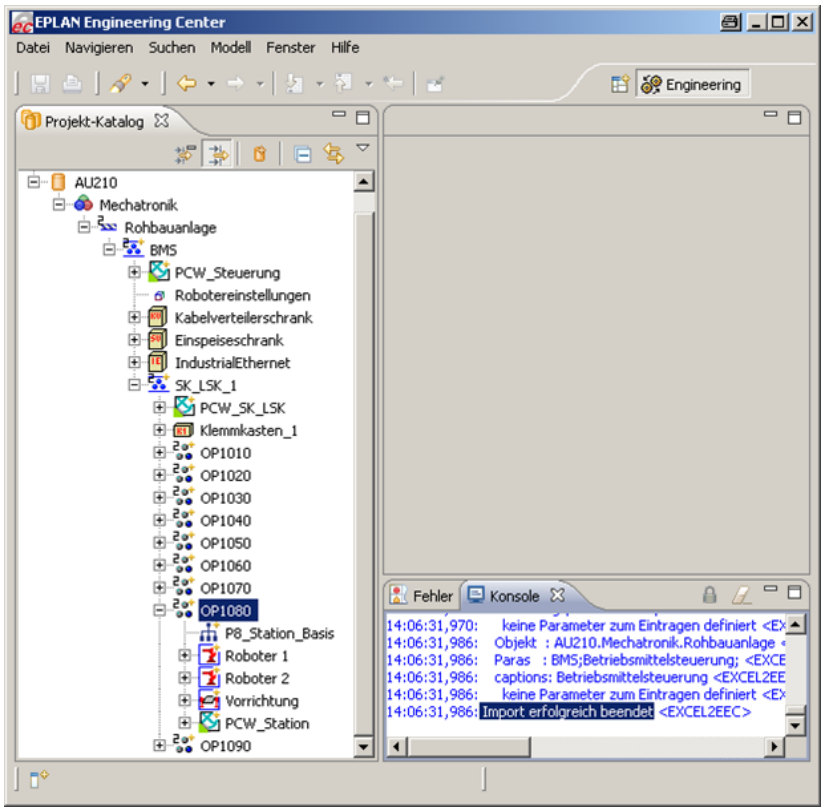


Figure 6.14.: Import of the body shop production line structure in the EEC

Changes in the core design in the EEC such as deleting, creating or renaming data are possible. In this example, operation 1090 with its robot was deleted.
When all changes have been made, the EEC project is exported to its native XML format which can be imported by the ProcessExplorer. Again the three steps are performed: first, the data is read; second, the relevant information is filtered; and third, the data is transformed according to the agreed upon naming conventions.

When the core design data project, based on the EEC data (including the changes), is saved to the same location where the original project is saved, a merging dialogue

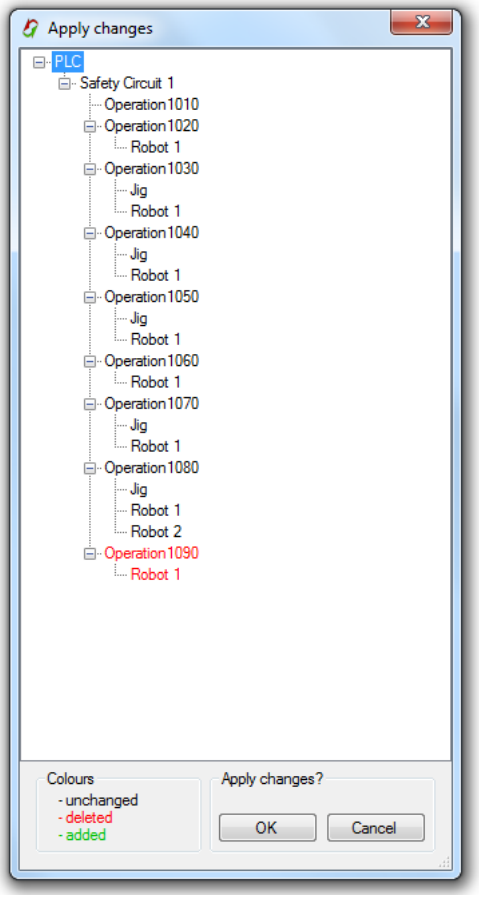


Figure 6.15.: Dialogue window for merging two ProcessExplorer projects

window appears to the user as shown in Figure 6.15. This window assists the user in deciding which difference/change to accept and which to discard. Elements that have not been changed are coloured in black, while new elements are green and deleted elements are red. Any renamed or rearranged elements appear as deleted in the old position or with the old name and as a new element in the new position or with a new name. An element cannot simply be changed; there are only deleted and new elements. The advantage of this approach is that the complexity of data merging is reduced and thus the merging process is more transparent to the user.

At this point the user can undo changes made but as soon as they are applied, the data on the network device is overwritten and all other engineering disciplines involved continue their work based on the changed core design data.

6.2.3. Exchange of Detailed Design Data

This case includes one engineering discipline adding a detailed design data module to the core design data which is accessed and further processed by another engineering discipline. The tasks belonging to the case as well as the engineering disciplines involved and the software tools employed are shown in Figure 6.16.

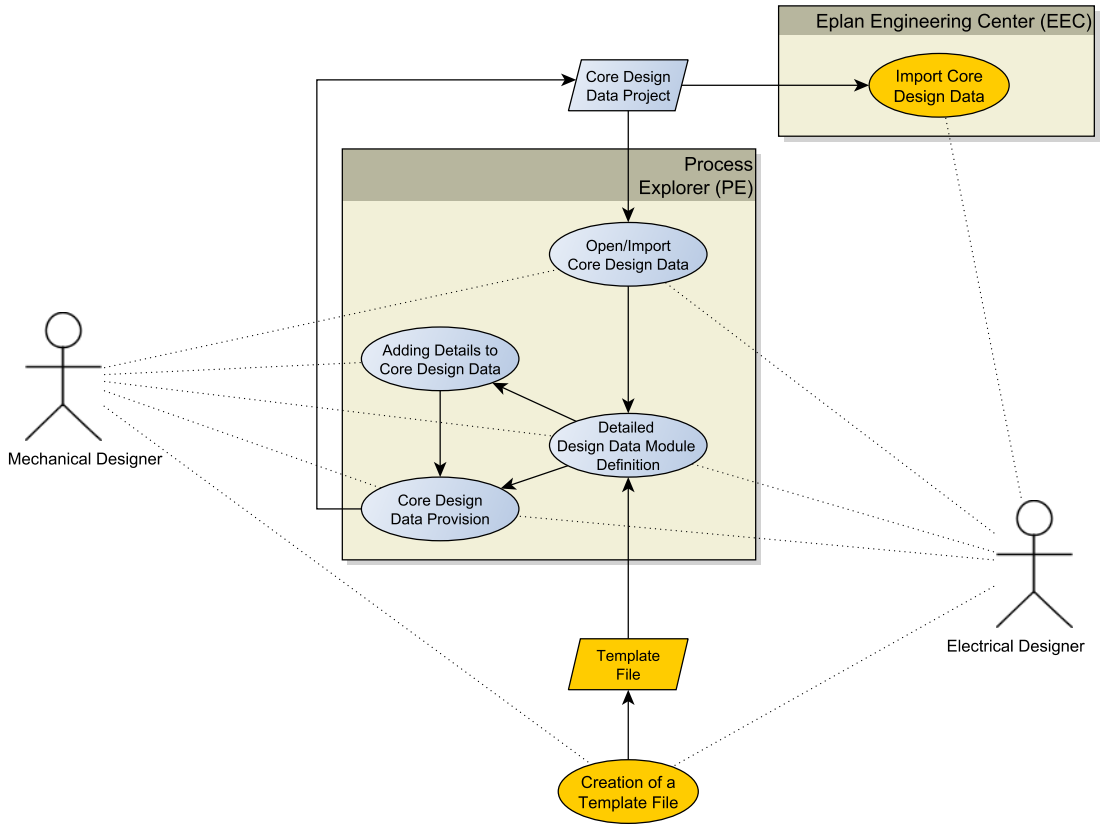


Figure 6.16.: Use case: exchange of detailed design data

The use case begins with the definition of a template file for a detailed design data module. It specifies the format of the detailed design data module and it is created in cooperation with the engineering disciplines which exchange the information, in this case: mechanical and electrical engineering. The file setup and format is defined according to the import and export capacities of the software tools producing and using the stored information. For this use case, the template file contains an empty spreadsheet which already contains the column headings and the definition of the data types stored in the specific cells. The file is prepared to include data of the automated components of a jig as specified in its 3D design by the mechanical designer. These automated components

include pneumatic cylinders, valves, position switches, proximity switches, etc. The mechanical designer, who is responsible for the detailed design data module, produces this information. The engineer responsible for a detailed design data module creates a new detailed design data module with the aid of the ProcessExplorer. For this it provides a configuration dialogue window as shown in Figure 6.17.

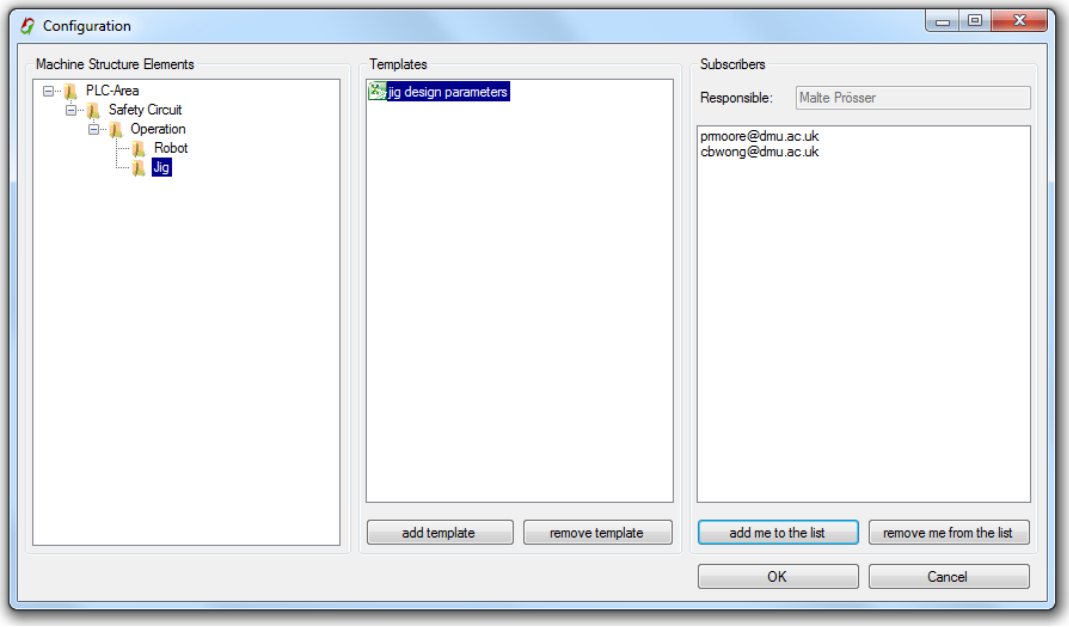
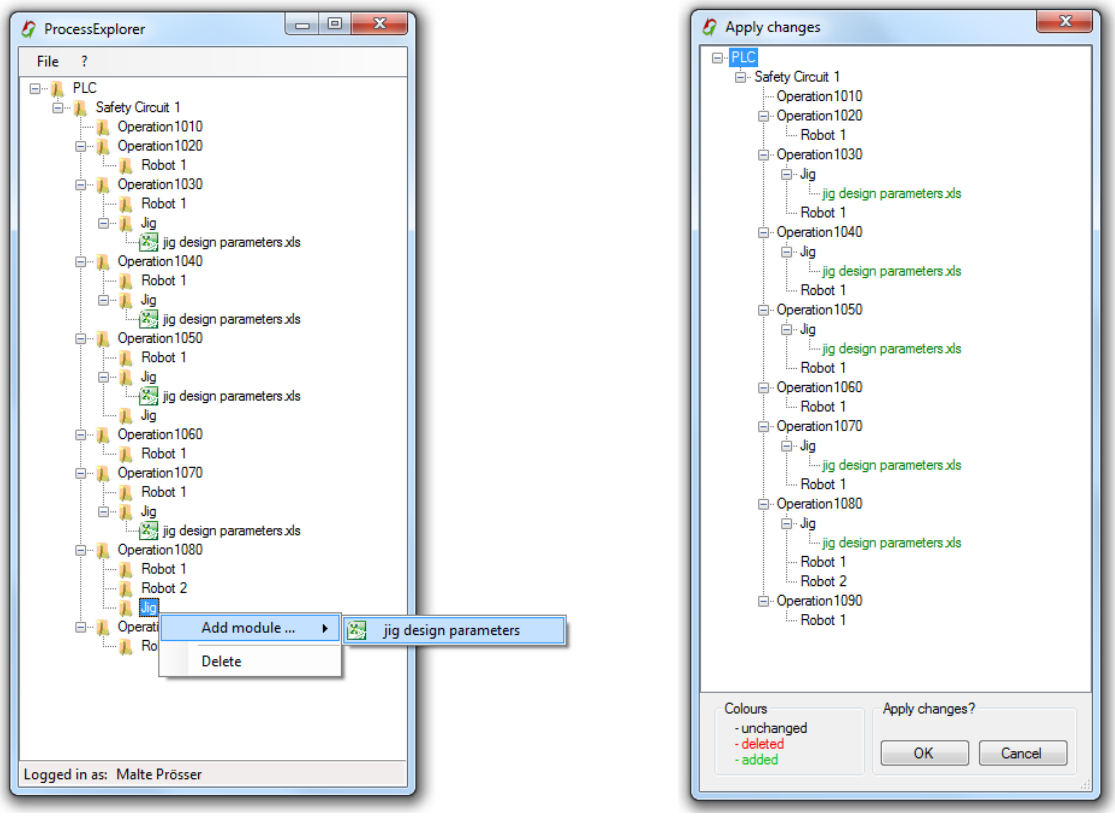


Figure 6.17.: Configuration dialogue window for the creation of a new detailed design data module

The module is connected with the core design data element *jig* in the left part of the dialogue, since jigs are further specified by the content of the file. The creation of a core design data module includes the upload of the template file which can be carried out in the middle part of the dialogue. The ProcessExplorer stores the module definition in a core design data project. Once the creation of a detailed design data module is finished and the core design data project is saved, it can be accessed by the other disciplines involved in the engineering design process.

For this use case, the electrical engineering designer accesses the core design data project and opens the configuration dialogue for the detailed design data module. There he can subscribe to the detailed design data module, thus indicating that he is going to set up with his work on the information stored in the module. As such he will be automatically informed by the ProcessDesigner when a new module is added to the core design data or when a present module is deleted or changed.



(a) Adding a detailed design data module to a core design data element

(b) Apply changes window

Figure 6.18.: ProcessExplorer: Exchange of Detailed Design Data

After the electrical designer has subscribed to the module, the mechanical designer adds a new module to the core design data by opening the project and either right clicking or dragging and dropping a filled out template file on the specific jig in the main window of the ProcessExplorer as is shown in Figure 6.18(a).

As soon as the ProcessExplorer project is saved, the merge dialogue appears indicating the changes made in the same way as it appears when changes in the structure are made (see Figure 6.18(b)). As previously mentioned new detail design modules appear in green indicating that they have been added to the project, while removed modules are displayed red. The user can review and revise changes.

When the mechanical designer saves the project, the electrical designer receives an email, automatically sent by the ProcessDesigner, which informs him of the recently added detailed design data module. The electrical designer then imports the core design data project, including the detailed design data module in the EEC.

6.3. Summary

The characteristics of the implementation method applied during the case study can be summarised as follows.

The implementation uses present export interfaces or native file formats. The advantage is that presently employed software tools can be used in the future as well. There is no need to purchase new software tools and adapt present engineering design processes accordingly.

Storing the information of detailed design data modules in text-based file formats or spreadsheets allows the various engineering disciplines to keep track of the information exchanged because they are able to control the design of the file setup. The automated email transmission is an easy-to-control method to keep all involved engineers updated on the progress. Emails are as easy to handle as spreadsheets or text files and can be filtered by keywords allowing users to focus on the changes most relevant to them.

In order to formally define templates of detailed design modules for further automatic processing, the mechatronic engineering design process must be analysed with regard to three major aspects:

- the differentiation aspect: which information is relevant to whom
- the chronological aspect: when information is processed
- the technical aspect: in which form the processed information is stored

The results of this analysis provides not only a solid base for discussions on improving the process and slimming down interfaces but is also essential for setting up any further advanced systems integration solutions.

7. Evaluation and Assessment

In the previous chapter the new middle-in data modelling strategy was implemented in an industrial engineering design process. The efficiency and robustness compared with present bottom-up and top-down data modelling strategies are evaluated in this chapter by simulating and discussing its reaction on certain disturbances on the data exchange as they occur in industrial engineering design processes. According to these typical disturbances three test cases have been developed for comparing the characteristics of systems integration solutions based on the three different data modelling strategies. The three test cases are used to evaluate how well the systems integration solutions based on each of the three data modelling strategies are able to face the major challenges of data modelling.

7.1. Evaluation Test Cases

Three test cases were developed to evaluate how well the different data modelling strategies are able to deal with the major challenges of systems integration as identified in the literature review in Chapter 2. One of the identified major challenges is the handling of different nomenclatures and the assuring of data model consistency. Test case 1 tests how systems integration solutions based on each of the three different data modelling strategies is able to face this challenge.

The other two major challenges of systems integration are gaining broad user acceptance and developing feasible introduction strategies. These challenges very much depend on the ability of an integration solution to deal with changes in the data models and in the applied software tools. So test case 2 tests how the systems integration reacts to changes in the data model and test case 3 tests how it reacts on changing one of the applied software tools. The purposes of each test case is summarised in Table 7.1.

	Maintenance of data model consistancy	Fulfilment of multidisciplinary requirements	Introduction strategy to industrial processes
Test Case 1: Causing Inconsistencies	X		
Test Case 2: Changing Data Models		X	X
Test Case 3: Changing Involved Software Tools		X	X

Table 7.1.: Test cases for testing the ability of systems integration solutions to deal with major challenges in data modelling

7.1.1. Test Case 1: Causing Inconsistencies

Inconsistencies occur when a set of information stored in a data model is changed but a corresponding set stored in the same data model is not adapted accordingly due to a missing link between the two sets. Such a link is typically a rule or a constraint. Changes in data likely occur during industrial engineering design processes as described above in sections 6.2.2 and 6.2.3.

The advantage of a top-down designed data model is that the consistency of the data is one of the major objectives of its design. It means that every single piece of information is only stored at one certain place in the data model and no information is stored at different locations where they are not linked for a simultaneous update. In a data model which has been designed bottom-up, it is very difficult to oversee the data model as a whole. So when one data model after the other is integrated, the same information is likely to be stored in several ways without being linked together, implying that data is likely to become inconsistent.

The separation into core design data relevant for all involved engineers and detailed design data modules relevant for a few as the central approach of the middle-in data modelling strategy also has a strong impact on the consistency of the data. Since the core design data is modelled top-down, its consistency is assured during the modelling phase.
The detailed design data modules are designed bottom-up. However, the amount of data stored in a detailed design data module is very limited since it only contains

detailed design information exchanged at a certain point of the design process among a few of the engineers. So, the links and dependencies between the data stored in a detailed design data module and the core design data are very limited. Thus, they can be overseen and managed by the single engineer, the one who is responsible for the detailed design data module.

To sum up, data stored in top-down design data models is most consistent because links and dependencies between the information are considered in the early data modelling phase. In contrast to this, data stored in bottom-up designed data models is least consistent because links and dependencies between the information are hard to oversee and thus hard to manage. Data stored in a middle-in designed data model are not as consistent as data stored in a top-down designed data model because only links and dependencies of data stored in the core design data are considered in the early modelling phase. Nevertheless, the risk of data becoming inconsistent is not as high for middle-in designed data models when compared with the bottom-up approach because the links and dependencies of data stored in detailed design data modules are very limited and can be overseen and managed by one single engineer.

7.1.2. Test Case 2: Changing Data Models

Changes in the data model likely occur during interdisciplinary engineering design processes meaning that new objects are added to the data model or that present objects are changed or deleted. These changes are typically due to technical progress, meaning that certain elements are added with advanced features which can be parametrerized by additional attributes. Another reason for these changes may be improvements in the data model due to experience from previous design projects or issues which are simply forgotten to consider in earlier design phases. This test case simulates how systems integration solutions based on the different data modelling strategies react to these kinds of changes.

The top-down data modelling strategy does not involve undertaking changes in the data model when it is already completed. A change in the top-down designed data model requires an assessment of the impact of this change in every software tool involved and usually it is necessary to change at least one of them. This is because all engineers and software tools involved process the whole amount of data independent of whether it is relevant for a certain discipline because the top-down data modelling strategy does not distinguish between data relevant for all engineering disciplines and data relevant to only a few.

Hence, top-down data models hardly exist in industrial environments, even though many systems integration activities start with the definition of a top-down data model. However, as soon as the first changes in the data model are required, these systems integration activities acquire an increasingly bottom-up character.

Changes to a bottom-up data model are incorporated just as one would integrate a new engineering discipline to it. However, the impact of this change to all of the software tools involved needs to be assessed just as required for changing a top-down designed data model. This is also because, that designing a data model bottom-up does not involve distinguishing between data relevant for all engineering disciplines involved and data relevant for only a few.
Additionally, when undertaking changes in both top-down and bottom-up data models, links and dependencies of the changed data with the unchanged data can hardly be overseen. Thus, the integrity and consistency of the changed data cannot be guaranteed.

The assessment of the abilities of the middle-in data modelling strategy to handle changes in data models relates particularly to changes in the core design data and changes in the detailed design data modules.
The core design data is as difficult to change as is the case for any other top-down designed data model. However, the data model of the core design data is not subjected to many changes because it contains the data relevant for all engineering disciplines. This data is typically the inherent basic structure of the element which is to be designed. This basic structure is usually adopted from one design project to the other with no changes.
The data model of a detailed design data module is likely to be changed more often because it contains in-depth technical details which is usually enhanced from project to project. However, changes in the detailed design data modules do not impact the whole data exchange process but only the few engineering disciplines which exchange the respective module. So only these few engineering disciplines, which are typically not more than two, need to adapt their data modules and their software tools while the rest of the exchanged data remains untouched. The consistency of the changed data and its integrity with the rest of the exchanged data is assured by the engineering discipline involved. So, the modularised character of the middle-in data modelling strategy allows changes in the data model to be dealt with much better than top-down and bottom-up data modelling strategies.

7.1.3. Test Case 3: Changing Software Tools Involved

The third test case simulates how an integration solution based each of the three different data modelling strategies reacts on changing one of the software tools involved. Such a change of a software tool likely occurs in industrial engineering design processes because the engineering disciplines involved aim to apply best in class software tools which are best possibly suited to their specific needs.

In a systems integration environment based on a top-down data model, single software tools can only be replaced when the new tool also supports the complete top-down data model. Even though modern software tools with an object oriented architecture allow certain degrees of freedom in adapting the underlying data models, it is still almost impossible to realise the complete top-down data model. Adapting the top-down data model to the data model underlying the new software tool is not provided for in the top-down data modelling strategy because it intends to include all data in the early data modelling phase and to consider all links and dependencies in the data model in order to avoid redundancies and keep the data consistent.

The possibility to incorporate changes later in the design life cycle after the early design phase is one of the fundamental distinctions between the top-down and bottom-up data modelling strategies. A systems integration environment where the underlying data model is designed bottom-up is prepared to integrate one software tool after another. Exchanging a current software tool only differs from introducing a new software tool in that the data from the exchanged software tool needs to be extracted from the bottom-up data model before introducing the new software tool. The difficulties to oversee links and dependencies between the data also makes it difficult to extract data from the data model which related to the previous software tool.

Of course, the data from the previous software tool can just be left in the bottom-up data model. However, since the new software tool usually has similar tasks to fulfil, the underlying data is also very similar. So, when introducing the new data model to the top-down data model, similar information are likely to be stored in different parts of the data model without being linked together. This leads to inconsistencies. Moreover, data exchange becomes more inefficient because more information is stored and transferred than necessary.

Exchanging a software tool in a middle-in data modelling environment effects the detailed design data modules, because every detailed design data module which has been exchanged by the previous software tool needs to be adapted to the new one. However, every other detailed design module remains untouched and thus, data exchange between the unchanged software tools is still possible without restrictions.

In addition, the new software tool needs to be able to store and handle the core design data. However, it is usually possible to implement the core design data to every modern engineering tool because the core design data mainly defines naming conventions and hierarchical relations of elements which can be implement in almost every modern software tool with an object oriented architecture.

To sum up, replacing a software tool in a systems integration environment where the underlying data model is designed top-down requires the adaption of the software tool to the top-down data model. In an environment with a bottom-up data model, the data model is adapted while the software tool remains untouched. In contrast to this, when the underlying data model is designed with a middle-in strategy it is required to adapt both the software tool according to the core design data and the detailed design data modules according to the new software tools.

7.1.4. Evaluation Results

Every data modelling strategy has its strength and weaknesses when compared with each other as summarised in Table 7.2.

	Top-down	**Bottom-up**	**Middle-in**
Data Exchange Efficiency	low	low	high
Data Integrity and Consistency	high	low	middle
Flexibility to adapt further developments	low	high	middle

Table 7.2.: Evaluation results

The top-down modelling strategy has its strength in keeping data consistent because consistency is considered as one fundamental design principle. However, a top-down designed data model is very inflexible because incorporating changes is

not included in top-down data modelling. In contrast, flexibly adapting to changes is a central strength of bottom-up data modelling due to its step-by-step integration approach.

Nevertheless, both data modelling strategies do not distinguish between data relevant for all engineering disciplines involved and data relevant for only a few. Therefore, in a top-down or bottom-up designed integration environment the complete data set is always exchanged. So, the distinction of data according to its relevance to the involved engineers is a central strength of the middle-in data modelling making it the most efficient strategy. Moreover, the middle-in data modelling strategies combines the strengths of bottom-up and top-down strategies in terms of consistencies and flexibility. However, data modelled by applying the middle-in strategy cannot be consistently as good as data modelled top-down and as flexible as data modelled bottom-up.

The systems integration solution based on middle-in data modelling is broadly accepted by the engineers involved in the engineering design process of body shop production lines at the AUDI AG's tooling and manufacturing equipment division. The middle-in approach allows the engineers themselves to introduce, maintain and further develop the integration solution for enhanced data exchange. The integrated software environment rises the efficiency of their design work. So, middle-in data modelling is set to be the primary strategic approach for further systems integration in the mechatronic engineering design process of body shop production lines.

7.2. Discussions of the Implications from the Evaluation Results

This section discusses the implications of the case study results which have been presented in the previous section. This involves reviewing the abilities of a systems integration solution based on one of the three data modelling strategies to handle the major challenges in systems integration which are handling different nomenclatures, gaining broad user acceptance and developing feasible introduction strategies.

7.2.1. Towards Handling Different Nomenclatures

The top-down data modelling strategy has the shortcoming that not all variants in nomenclature as used by different engineering disciplines are considered. However, its strength is that it considers and preserves consistency as a key aspect of its modelling approach. In contrast to this, bottom-up data models consider various nomenclatures: however, bottom-up data models rapidly grow in complexity as each software system is added. Thus, dependencies and constraints in the stored information are hard to detect and, consistency can therefore be preserved only with great difficulty.

Applying the middle-in data modelling strategy combines the strengths of both bottom-up and top-down approaches and at the same time minimises the impact of the shortcomings. To define the core design data the nomenclatures of the disciplines involved need to be aligned. However, the definition of core design data limits the size of the agreed upon set of data since the core design data is only a part of the whole data exchanged during the process. Thus, the middle-in data modelling strategy reduces the workload necessary for aligning different nomenclatures to a minimum and agreements are struck more easily.

Since the core design data is part of the data models underlying all software tools applied by the engineering disciplines involved, this data can be easily synchronised because it can be transferred directly. No resources are needed to implement and maintain the transfer algorithms. The quality of the engineering design process is enhanced by a better flow of information, as every engineer is informed about changes to the fundamental structure of the production facility. Since all the engineering disciplines in the project are synchronised with this core data model, they can react to the change simultaneously without delay. No working time is wasted because an engineer always works on an updated version of the underlying structure.

Data which goes beyond the core design data is stored in detailed design data modules. Setting the nomenclature of data stored in detailed design data modules requires the agreement of only a few engineering disciplines, so agreement is more easily reached.

Approaching data modelling with a middle-in strategy offers a more balanced way to ensure the maintenance of data model consistency. This can be done by assigning clear responsibilities on the basis of regulated information updates. All engineers

involved in the design process are responsible for the core design data which is clearly separated from the detailed design data modules. Furthermore, the responsibility for these modules can be explicitly assigned to the engineers producing and using the information.

7.2.2. Towards Gaining Broad User Acceptance

A disadvantage of the top-down approach is that it is difficult to incorporate requirements which were not considered during early modelling. Of course, this has a negative impact on user acceptance. In many cases engineers need to change their applied software tool since it does not support the top-down data model. In contrast to this, bottom-up data modelling allows stepwise consideration of each discipline's requirements and software tools can be applied further on.
The middle-in data modelling strategy can be realised with implementation technology of any complexity. This includes simple implementation technology such as network shared devices, email or spreadsheets as shown in this case study. These simple implementation technologies should be familiar to the engineers from their daily work. This allows them to influence and control the systems integration process which should increase user acceptance.

Once a set of information stored in a detailed design module is agreed upon, it is possible that the software tools used by the engineers will be able to import the detailed design modules automatically with no need to carry out supplementary manual processes. Thus, the amount of work necessary to implement and maintain import scripts and transfer algorithms is kept to a minimum. Naturally, this also saves time in the creation process and enhances the quality of the design.

The detailed design data module facilitates communication between the engineers who are working together. They agree on how to transfer data, in what form and with which values. This supports the engineer's understanding of the cooperative design process, meaning the understanding of the form and level of detail of their deliverables as expected by engineers in the later steps of the process.

7.2.3. Towards Developing Feasible Introduction Strategies

Systems integration solutions based on a bottom-up data modelling strategy can be introduced in stages, considering requirements discipline by discipline and implementing the interfaces independently of each other. Top-down systems

integration does not allow a stepwise approach: it requires an adjustment of the engineering design process as a whole. Systems integration solutions based on a middle-in data modelling strategy can be introduced in steps. Starting with the definition of the core design data, additional disciplines can be integrated by defining new detailed design data modules.

The middle-in data modelling strategy works in the same way, regardless of whether the system contains one detailed design module or dozens of detailed design modules. It does not matter if only one type of detailed design data modules is exchanged between two designers or if dozens of modules are exchanged between various designers from all disciplines. The encapsulation of detailed design information into detailed design modules allows a step-by-step consideration of each discipline's requirements. Each module has an engineer responsible for its creation and a finite number of engineers who use the information contained in the module for their work.

It is clear to every subscriber who the source and the producer of the information is. The producer is responsible for the information produced, whether it is correct or not. Moreover, the producer of the information can find out who is going to build upon the information produced by checking the subscribers list. The process is, therefore, quite transparent and mistakes can be cleared up quickly. Because of the clear assignment of responsibilities no time is wasted on discussing who is responsible for a mistake. Moreover, information which engineers need is always readily available from a single source, the detailed design module. The users do not have to search through various sources for the information they need. This too saves time in the engineering design process.

Furthermore, the reasonable definition of detailed design modules allows engineers to work increasingly in parallel. Modules containing required information can be produced earlier in the process. The same applies to modules containing information which is required and can be produced later in the process. Thus, engineering disciplines can start working on the intermediate results instead of waiting for the final results.

7.3. Transferability of Evaluation Results

Setting up systems integration on a middle-in data modelling strategy is not limited to the engineering design process for body shop production lines. It can be applied

to a variety of other mechatronic engineering design processes which require that engineers and their software tools work together seamlessly. It as applicable to any other process where a set of core design data exists which is relevant to all engineering disciplines and when detailed design data modules are meant to be added to the core design data for further enriching the core data.

The middle-in data modelling strategy supports engineers to break down the systems integration process into manageable steps: core design data and detailed design. This exposes the fundamental tasks of systems integration for any complexity of the used implementation technology:

- separate data according to its relevance: which data is relevant to all involved stakeholders and which data is only relevant to a few?
- standardise and align the structures of data relevant to all involved stakeholders
- design the structures of data relevant to a few stakeholders according to the conditions and possibilities of the existing software tools in use and producing the information to process the data with as little manual intervention as possible

These are the fundamental tasks of system integration independent of the complexity of the applied implementation technology. Based on this, systems integration can be realised using rudimentary implementation technologies which allow an immediate response to the diverse and sometimes contradictory requirements of the various engineering disciplines. The engineers themselves can introduce and adapt the integration solution specially tailored to their individual needs, which makes the integration solution the best possible fit to the engineering design process.

8. Conclusions and Future Work

This chapter draws the dissertation to an end and summarises the specific contributions to knowledge resulting from this research. In addition, potential directions for future research work are discussed.

8.1. General Conclusions of the Thesis

The growing complexity of manufacturing systems requires close cooperation between the engineering disciplines and the software tools they use. Consequently, systems integration in mechatronic engineering design processes for manufacturing systems must be able to react to product and process driven requirements. In addition, systems integration makes it possible for the mechatronic engineering disciplines to work increasingly simultaneously since the designers can work on intermediate results and recently updated data with no need to wait until final results are available, thus shortening and increasing the efficiency of the process.

A new middle-in data modelling strategy was presented which enables efficient systems integration in a mechatronic engineering design process for manufacturing systems. It is more efficient because it combines the advantages of present top-down and bottom-up data modelling strategies. The middle-in data modelling approach considers data consistency and at the same time allows introductions to current industrial engineering design processes in steps and is still flexible enough to be adapted to a changing environment.

Integration solutions based on the middle-in data modelling strategy can be implemented using technology of any complexity. It can be used with simple implementation technologies which can be set up and maintained by the engineers themselves. Moreover, it is not necessary to migrate the whole engineering design process to an integrated tool suite; engineers can keep using best in class software tools matching their individual needs and requirements.

The middle-in data modelling strategy was successfully applied to systems integration in the mechatronic engineering design process of automotive body shop production lines in a case study. Software tools were integrated which were previously unable to exchange data and the involved engineers implemented the integration solution themselves, using simple technologies. The case study confirmed that the success of an integration solution is independent of the chosen implementation technology when approaching systems integration with a middle-in data modelling strategy. This is due to the fact that middle-in data modelling divides systems integration into manageable subtasks with distinct interfaces. However, these qualitative benefits are difficult to quantify. This is due to the uniqueness of single batch and special purpose manufacturing systems and their equally unique engineering design processes which are difficult to compare quantitatively.

The insights gained from the case study about the benefits of the middle-in data modelling strategy are not limited to the mechatronic engineering design process of body shop production lines. A general description of the middle-in data modelling strategy by means of set theory confirms its general applicability to a broad range of systems integration tasks in a variety of industries.

Based on current trends, it can be expected that the number of heterogeneous software systems will increase and the necessity to distribute data will thus be even greater. Therefore, it is expected that the importance of data modelling will further increase and a structured approach to aligning heterogeneous nomenclatures will be a key enabling factor for efficient systems integration. Additionally, it will be even more important to integrate software systems in steps and flexibly incorporate new or changed software. Engineers will have a better overview of the whole engineering process than external system integrators. Thus, it will be more important to set up on system integration technologies well known by the engineers to benefit from their process knowledge and experience.

8.2. Contributions to Knowledge

The specific contributions to knowledge resulting from this research are as follows:

Taking present integration solutions as a starting point, the first contribution is the identification of the three major challenges in data modelling during mechatronic engineering design of manufacturing systems. The first challenge is

the maintenance of data model consistency in terms of avoiding the storage of the same data at multiple positions as well as contradictory data in the data model. The second challenge is responding to diverse and in part contradictory user requirements when summarising data from different engineering disciplines in a single data model. Finally, the third challenge is introducing a systems integration solution to current industrial engineering design processes. This usually requires a stepwise introduction taking the use of current software tools into account.

The second contribution is the classification of data modelling strategies and their comparison with regard to meeting the challenges identified. Current approaches are either bottom-up or top-down data modelling strategies. The advantage of the bottom-up approach is its response to user requirements and ability to be introduced to an industrial engineering process in steps. However, bottom-up data models rapidly grow in complexity as more software systems are integrated, and then consistency of the data is difficult to maintain. This is the major advantage of the top-down modelling approach because top-down data models are designed to be consistent.

The derivation of the middle-in data modelling strategy which combines the advantages of current modelling strategies is the third contribution. The middle-in data modelling strategy divides the data to be exchanged into core design data and detailed design data modules. The core design data is designed top-down; thus, this data is intended to be consistent for all engineering disciplines involved in the design process. The detailed design data modules which contain data relevant for a few (typically two) engineering disciplines is designed bottom-up. This allows the involved engineers to incorporate their individual requirements into the data model and, additionally, to incorporate it into the engineering design process in steps.

The fourth contribution is presenting the details of a successful application of the middle-in data modelling strategy to an industrial engineering process. The middle-in data modelling strategy was successfully applied to the mechatronic engineering design process for body shop production lines for the automotive industry in a case study. It confirms that the middle-in data modelling strategy alleviates the disadvantages of bottom-up and top-down approaches.

8.3. Future Work and Outlook

Future work should enhance systems integration in industrial environments on the operational level and expand its study on the academic level.
The operational level includes the expansion of the middle-in data modelling strategy to fields other than the creation process of manufacturing systems. For example, the increasing interconnection among these systems will make it possible to constantly collect data from the shop floor and compare it with the initial design and thus to constantly improve manufacturing design. Of course, the data collected from the shop floor will be as heterogeneously stored as engineering data accruing during the mechatronic design process. Thus, the need to integrate information remains the same and the constraints as observed for systems integration in mechatronic design processes applies for integrating data from the shop floor. Applying bottom-up and top-down strategies for modelling data from the shop floor should also have similar limitations in sustaining the consistency of the data and in fulfilling contradictory requirements on the data model. Thus, the middle-in data modelling strategy is also a promising approach for integrating data from the shop floor, and this is a potential field for future research work.

A promising field for future work is investigating new ways to evaluate the impact of different systems integration approaches more quantitatively. It is very difficult to compare performance indicators of different engineering design projects of complex and single batch facilities like body shop production lines. Usually, such facilities have long lead-times of several years. So, the facilities develop due to enhanced technology from one project to another. Moreover, the software tools used by the engineers for their design work also evolve and the team of involved engineers likely changes as well. These factors greatly influence the captured performance indicators making it difficult to quantitatively evaluate the contribution of a new systems integration solution to an enhanced engineering design process.

Another area for future work on the operational level is enhancing the education of engineers in systems integration. Simple integration technologies such as spread sheets have been successfully applied in this research to allow the engineers themselves to design the data modules exchanged for systems integration. Based on the results of this research, it is seems essential to involve the users themselves in the systems integration process to ensure that it suits their specific needs. However, because the integration possibilities of spread sheets are limited in terms of data volume and complexity, and advanced integration technologies such as

databases, XML-files or message-based integration require profound IT knowledge and skills, further research is required to develop easy-to-use advanced integration technologies.
Moreover, teaching systems integration should be part of future manufacturing engineering education or new engineering disciplines such as computer science of manufacturing systems should be further established.

There is work to be done at the academic level to make systems integration more quantifiable, since the results of this research are rather qualitative. The complexity of the systems integration task could be determined by introducing appropriate indicators. These indicators would make it possible to compare several tasks with each other and suitable systems integration strategies could be determined based on the indicators. Therefore, it would be necessary to identify more quantifiable case studies which are more comparable with each other than body shop production line projects.

Another area of future academic work is based on the mathematical description of the middle-in data modelling strategy with the aid of set theory in Chapter 4.4. This theoretical foundation of the middle-in data modelling strategy could be further elaborated. It might be possible to design and implement an algorithm which automatically determines the core design data and detailed design data modules from several data sets.

Reference List

Adolfsson, J., Ng, A., Olofsgård, P., Moore, P., Pu, J. and Wong, C. (2002), 'Design and simulation of component-based manufacturing machine systems', *Mechatronics* **12**(9-10), 1239–1258.

Anderl, R. and Trippner, D. (2000), *STEP–Standard for the Exchange of Product Model Data*, Teubner.

Arnaud, R. and Barnes, M. (2006), *COLLADA*, Peters Wellesley, Mass.

Automation ML consortium (2013), 'Whitepaper AutomationML Part 1 – Architecture and general requirements', `https://www.automationml.org/o.red/uploads/dateien/1375858464-AutomationML%20Whitepaper%20Part%201%20-%20AutomationML%20Architecture%20V2.2.pdf`. Accessed 29 September 2013.

Bakis, N., Aouad, G. and Kagioglou, M. (2007), 'Towards distributed product data sharing environments – Progress so far and future challenges', *Automation in Construction* **16**(5), 586–595.

Bergert, M., Diedrich, C., Kiefer, J. and Bar, T. (2007), Automated plc software generation based on standardized digital process information, *in* 'Emerging Technologies & Factory Automation, 2007. ETFA. IEEE Conference on', pp. 352–359.

Biffl, S., Schatten, A. and Zoitl, A. (2009), Integration of heterogeneous engineering environments for the automation systems lifecycle, *in* 'Industrial Informatics, 7th International Conference on', IEEE, pp. 576–581.

Brandt, S. C., Morbach, J., Miatidis, M., Theißen, M., Jarke, M. and Marquardt, W. (2008), 'An ontology-based approach to knowledge management in design processes', *Computers & Chemical Engineering* **32**(1), 320–342.

Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E. and Yergeau, F. (2008), Extensible Markup Language (XML) 1.0, Recommendation 5, World Wide Web Consortium (W3C) Recommendation, `http://www.w3.org/TR/2008/REC-xml-20081126/`.

Brière-Côté, A., Rivest, L. and Desrochers, A. (2010), 'Adaptive generic product structure modelling for design reuse in engineer-to-order products', *Computers in Industry* **61**(1), 53–65.

Chappell, D. (2004), *Enterprise service bus*, O'Reilly Media, Inc.

Chen, Y. (2010), 'Knowledge integration and sharing for collaborative molding product design and process development', *Computers in Industry* **61**(7), 659–675.

Choi, S. and Yoon, T. (2010), 'XML-based neutral file and PLM integrator for PPR information exchange between heterogeneous PLM systems', *International Journal of Computer Integrated Manufacturing* **23**(3), 216–228.

Chryssolouris, G., Mavrikios, D., Papakostas, N., Mourtzis, D., Michalos, G. and Georgoulias, K. (2009), 'Digital manufacturing: history, perspectives, and outlook', *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **223**(5), 451–462.

Conrad, S., Hasselbring, W., Koschel, A. and Tritsch, R. (2006), *Enterprise Application Integration*, Elsevier, Spektrum Akad. Verl.

Cutkosky, M., Engelmore, R., Fikes, R., Genesereth, M., Gruber, T., Mark, W., Tenenbaum, J. and Weber, J. (1993), 'PACT: An experiment in integrating concurrent engineering systems', *Computer* **26**(1), 28–37.

Dassault Systèmes (2013), 'CATIA - 3D Systems Shape Mechanical and Equipement Virtual Design', `http://www.3ds.com/products/catia/`. Accessed 7 September 2013.

Drath, R. (2010), *Datenaustausch in der Anlagenplanung mit AutomationML*, Springer Berlin Heidelberg.

Drath, R. and Barth, M. (2011), Concept for interoperability between independent engineering tools of heterogeneous disciplines, *in* 'Emerging Technologies & Factory Automation, International Conference on', IEEE, pp. 1–8.

Drath, R., Luder, A., Peschke, J. and Hundt, L. (2008), AutomationML-the glue for seamless automation engineering, *in* 'Emerging Technologies and Factory Automation, International Conference on', IEEE, pp. 616–623.

Drath, R. and Miegel, V. (2009), 'AutomationML verbindet Werkzeuge der Anlagenplanung', *Automatisierungstechnische Praxis* **7**, 34–39.

EPLAN Software & Service (2013, *a*), 'EPLAN Electric P8', `http://www.eplan.de/en/solutions/electrical-engineering/eplan-electric-p8/`. Accessed 9 September 2013.

EPLAN Software & Service (2013, *b*), 'EPLAN Engineering Center', `http://www.eplan.de/en/solutions/mechatronic/eplan-engineering-center/`. Accessed 9 September 2013.

EPLAN Software & Service (2013, *c*), 'EPLAN Fluid', `http://www.eplan.de/en/solutions/fluid-power-engineering/eplan-fluid/`. Accessed 9 September 2013.

ESPRIT Consortium AMICE (1993), *CIMOSA: Open System Architecture for CIM*, 2nd edn, Springer-Verlag.

Fenton, J. (1998), *Handbook of automotive body construction and design analysis*, Professional Engineering Publishing Limited.

Gruber, T. R. et al. (1993), 'A translation approach to portable ontology specifications', *Knowledge acquisition* **5**(2), 199–220.

Haq, I., Monfared, R., Harrison, R., Lee, L. and West, A. (2010), 'A new vision for the automation systems engineering for automotive powertrain assembly', *International Journal of Computer Integrated Manufacturing* **23**(4), 308–324.

Hohpe, G. (2002), Enterprise integration patterns, *in* 'Conference on Pattern Language of Programs', Addison-Wesley Reading, MA.

Hohpe, G. and Woolf, B. (2004), *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*, Addison-Wesley Professional.

IEC 61131-3 (2003).

Khilwani, N., Harding, J. and Choudhary, A. (2009), 'Semantic web in manufacturing', *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **223**(7), 905–924.

Kiefer, J., Baer, T. and Bley, H. (2006), Mechatronic-oriented engineering of manufacturing systems taking the example of the body shop, *in* 'Proceedings of the 13 thInternationsal Conference on Life Cycle Engineering, Leuven'.

Lüder, A., Foehr, L., Wagner, T., Zaddach, J. and Holm, T. (2010), Manufacturing system engineering with mechatronical units, *in* 'Emerging Technologies and Factory Automation, International Conference on', IEEE, pp. 1–8.

Mangold, C., Rantzau, R. and Mitschang, B. (2003), Föderal: Management of engineering data using a semistructured data model, *in* 'ICEIS 2003 - Information Systems Analysis and Specifcation', pp. 382–389.

Mascardi, V., Locoro, A. and Rosso, P. (2010), 'Automatic ontology matching via upper ontologies: A systematic evaluation', *Knowledge and Data Engineering, IEEE Transactions on* **22**(5), 609–623.

McLean, C., Mitchell, M. and Barkmeyer, E. (1983), 'A computer architecture for small-batch manufacturing', *IEEE Spectrum* **20**(5), 59–64.

MEDEIA Consortium (2008), State-Of-The-Art and Technology Review, Technical report, PROFACTOR GmbH.

Mewes & Partner GmbH (2013), 'WinMOD – Real-Time Simulation Center for Automation', `http://www.winmod.de/en/`. Accessed 9 September 2013.

Moser, T. and Biffl, S. (2010), Semantic tool interoperability for engineering manufacturing systems, *in* 'Emerging Technologies and Factory Automation, International Conference on', IEEE, pp. 1–8.

Moser, T., Biffl, S., Sunindyo, W. and Winkler, D. (2010), Integrating production automation expert knowledge across engineering stakeholder domains, *in* 'Complex, Intelligent and Software Intensive Systems, International Conference on', IEEE, pp. 352–359.

Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R. et al. (1991), 'Enabling technology for knowledge sharing', *AI magazine* **12**(3), 36.

Ng, H. C. (2003), An Integrated Design, Simulation and Programming Environment for Modular Manufacturing Machine Systems, PhD thesis, De Montfort University.

PHOENIX CONTACT (2013), 'PC WORX', `http://www.phoenixcontact.com/automation/32131_31906.htm`. Accessed 18 September 2013.

PLCopen Technical Committee 6 (2009), XML Formats for IEC 61131-3, Technical report, PLCOpen.

Pullan, T., Bhasi, M. and Madhu, G. (2010), 'Application of concurrent engineering in manufacturing industry', *International Journal of Computer Integrated Manufacturing* **23**(5), 425–440.

Qi, Z., Schafer, C. and Klemm, P. (2006), Interdisciplinary Data Exchange in the Development of Assembly Systems, *in* '2006 IEEE International Conference on Industrial Informatics', pp. 542–547.

RÜCKER EKS GmbH (2013), 'INVISION – Power of Simulation', `http://www.invision-simulation.de/`. Accessed 18 September 2013.

Reuter, A., Kircher, C. and Verl, A. (2010), 'Manufacturer-independent mechatronic information model for control systems', *Production Engineering* **4**(2), 165–173.

Schafer, W. and Wehrheim, H. (2007), The challenges of building advanced mechatronic systems, *in* 'Future of Software Engineering', IEEE Computer Society, pp. 72–84.

Scheer, A. (1990), *CIM Computer Integrated Manufacturing: Computer Integrated Manufacturing = der computergesteuerte Industriebetrieb*, Springer.

Scheer, A. (2000), *ARIS-business process modeling*, Springer.

Schleipen, M., Drath, R. and Sauer, O. (2008), The system-independent data exchange format caex for supporting an automatic configuration of a production monitoring and control system, *in* 'Industrial Electronics, International Symposium on', IEEE, pp. 1786–1791.

Schuh, G., Rozenfeld, H., Assmus, D. and Zancul, E. (2008), 'Process oriented framework to support plm implementation', *Computers in Industry* **59**(2), 210–218.

Shen, W., Hao, Q., Mak, H., Neelamkavil, J., Xie, H. and Dickinson, J. (2008), Systems integration and collaboration in construction: a review, *in* 'Computer Supported Cooperative Work in Design, 12th International Conference on', IEEE, pp. 11–22.

Shvaiko, P. and Euzenat, J. (2008), 'Ten challenges for ontology matching', *On the Move to Meaningful Internet Systems* pp. 1164–1182.

Siemens PLM Software (2013, *a*), 'Defining PLM', `http://www.plm.automation.siemens.com/en_gb/plm/definition/`. Accessed 18 September 2013.

Siemens PLM Software (2013, *b*), 'Process Designer: Assembly Planning and Validation', `http://www.plm.automation.siemens.com/en_us/products/tecnomatix/assembly_planning/process_designer.shtml`. Accessed 18 September 2013.

Siemens PLM Software (2013, *c*), 'Robcad: Robotics and Automation Planning', `http://www.plm.automation.siemens.com/en_us/products/tecnomatix/robotics_automation/robcad/index.shtml`. Accessed 18 September 2013.

Siemens PLM Software (2013, *d*), 'Teamcenter: Product Lifecycle Management', `http://www.plm.automation.siemens.com/en_us/products/teamcenter/index.shtml`. Accessed 18 September 2013.

Smith, B. M. (1983), 'IGES: a key to CAD/CAM systems integration', *IEEE Computer Graphics and Applications* **3**(8), 78–83.

Song, I., Yang, J., Jo, H. and Choi, S. (2009), 'Development of a lightweight cae middleware for cae data exchange', *International Journal of Computer Integrated Manufacturing* **22**(9), 823–835.

Stuckenschmidt, H. and Van Harmelen, F. (2005), *Information sharing on the semantic web*, Springer-Verlag New York Inc.

Tursi, A., Panetto, H., Morel, G. and Dassisti, M. (2009), 'Ontological approach for products-centric information system interoperability in networked manufacturing enterprises', *Annual Reviews in Control* **33**(2), 238–245.

Waltersdorfer, F., Moser, T., Zoitl, A. and Biffl, S. (2010), Version management and conflict detection across heterogeneous engineering data models, *in* 'Industrial Informatics, 8th International Conference on', IEEE, pp. 928–935.

Zhang, W., Yin, J., Lin, L. and Zhu, T. (2009), 'Towards a general ontology of multidisciplinary collaborative design for semantic web applications', *International Journal of Computer Integrated Manufacturing* **22**(12), 1144–1153.

A. Appendix

A.1. List of Publications

Partial results of this research have been published in following publications:

- M. Prösser, P. Moore, X. Chen, C.-B. Wong, U. Schmidt (2013), 'A new approach towards systems integration within the mechatronic engineering design process of manufacturing systems', *International Journal of Computer Integrated Manufacturing*, **26**(8):806-815
- M. Prösser, P. Moore, C.-B. Wong, U. Schmidt (2010), 'A concept for the Exchange of Engineering Data during Mechatronics Design Processes for Manufacturing Systems for the Automotive Industry', *Proceedings of the 12th Mechatronics Forum Biennial International Conference*, Zurich
- U. Schmidt, M. Prösser (2010), 'Ein neues Konzept zum Datenaustausch im mechatronischen Konstruktionsprozess von Fertigungsanlagen für die Automobilindustrie' (A new concept of data exchange during mechatronic engineering design of manufacturing systems for the automotive industry), *IAF Forschungsbericht*, Ingolstadt University of Applied Sciences
- U. Schmidt, M. Prösser (2009), 'Baukastenbasiertes Prozessmodell für Erstellung und kontinuierliche Weiterentwicklung von Karosseriebauanlagen' (Component-based process model for design and continuous improvement of body shop production lines), *IAF Forschungsbericht*, Ingolstadt University of Applied Sciences

A.2. Translations

Figure A.1.: Translation of Figure 4.2: Comparison of ProcessDesigner (left) and EEC (right) structure of a Resource Control

Figure A.2.: Translation of Figure 4.3: Comparison of ProcessDesigner (left) and EEC (right) structure of a Safety Circuit